Graphics • BASIC • C • Modula-2 • Pascal • COBOL

988
NDEX
E PAGE 7

The Resource Journal for IBM Micro Programmers

PROGRAMMER'S Journal

JANUARY/FEBRUARY 1989
Volume 7.1
$5.00 Canada $6.25

Graphics

VGA Programming in 256 Colors

Graphics Coprocessors from TI and INTEL

More Windows Programming

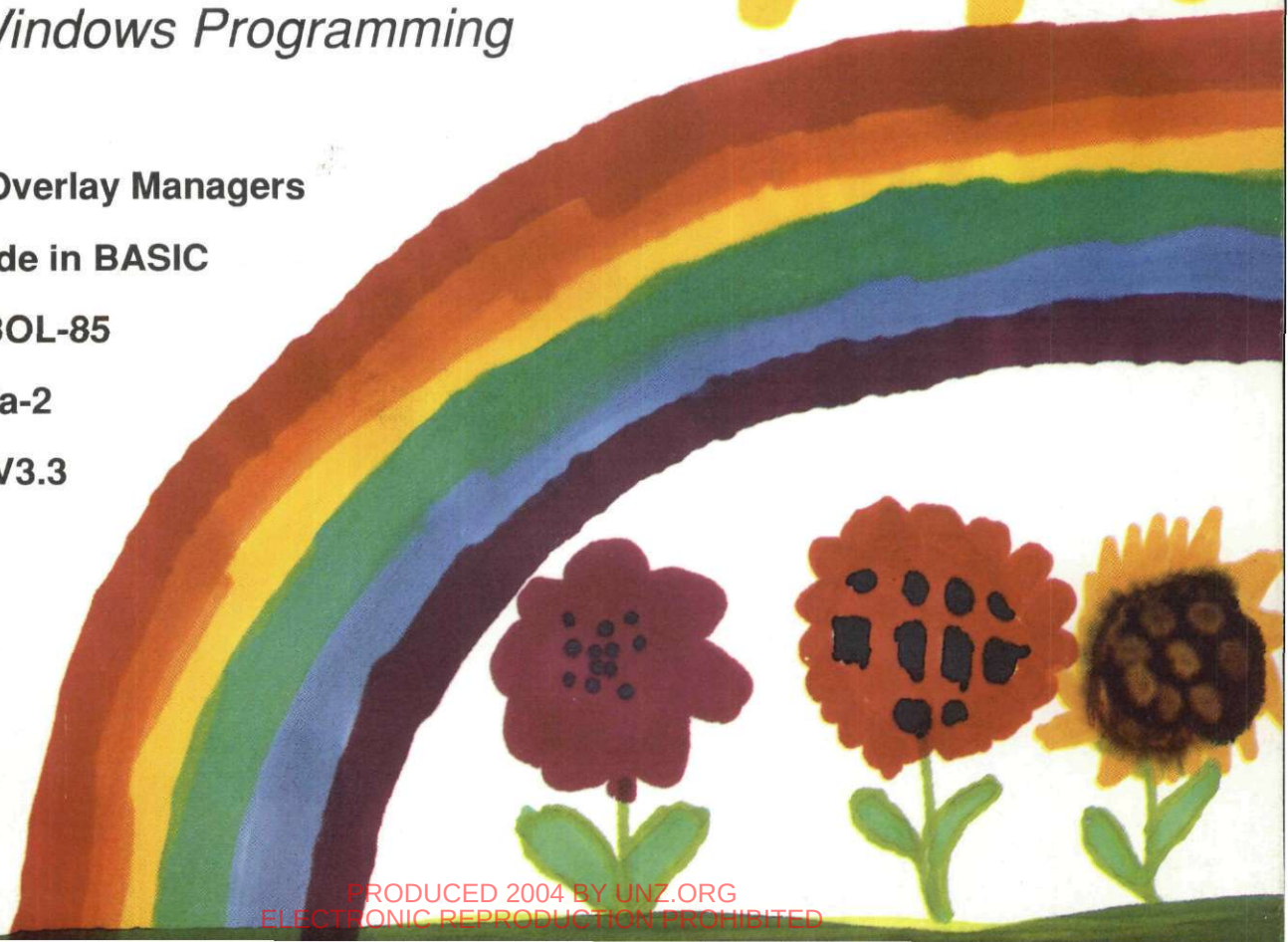
PASCAL Overlay Managers

Faster Code in BASIC

ANSI COBOL-85

JPI Modula-2

Lattice C V3.3



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MS-DOS $99

We call PolyAWK our "most flexible language" because it's a general purpose language that can replace a host of specialized tools or programs. You will still use your standard language (C, Pascal, Assembler, or other) or other modular language to develop applications, but you will write your own specialized development tools and programs with this versatile, simple and powerful language. Like thousands of others, you will soon find PolyAWK to be an indispensable part of your MS-DOS toolbox. If you don't have enough OS/2 development tools, now it's easy to write your own.

A True Implementation Under MS-DOS & OS/2

Bell Labs brought the world UNIX and C, and now professional programmers are discovering AWK. AWK was originally developed for UNIX by Alfred Aho, Richard Weinberger & Brian Kernighan of Bell Labs. Now PolyAWK gives MS-DOS & OS/2 programmers a true implementation of the valuable new programming tool. PolyAWK fully conforms to the AWK standard as defined by the original authors in their book, *The AWK Programming Language*.

A Pattern Matching Language

PolyAWK is a powerful pattern matching language for writing short programs to handle common text manipulation and data conversion tasks. A PolyAWK program consists of a sequence of patterns and actions that tell what to look for in the input data and what to do when it is found. PolyAWK searches a set of files for lines matched by any of the patterns. When a matching line is found, the corresponding action is performed. A pattern can select lines by boolean combinations of regular expressions and comparison operations on strings, numbers, fields, variables, and array elements; patterns may reference properties of the current input line or other program variables. Actions can perform arbitrary processing on selected lines. The action language looks like C, but there are no declarations and strings and numbers are built-in data types. You can have multiple input files and output files, regular expressions, user-defined functions, and nine other programs.

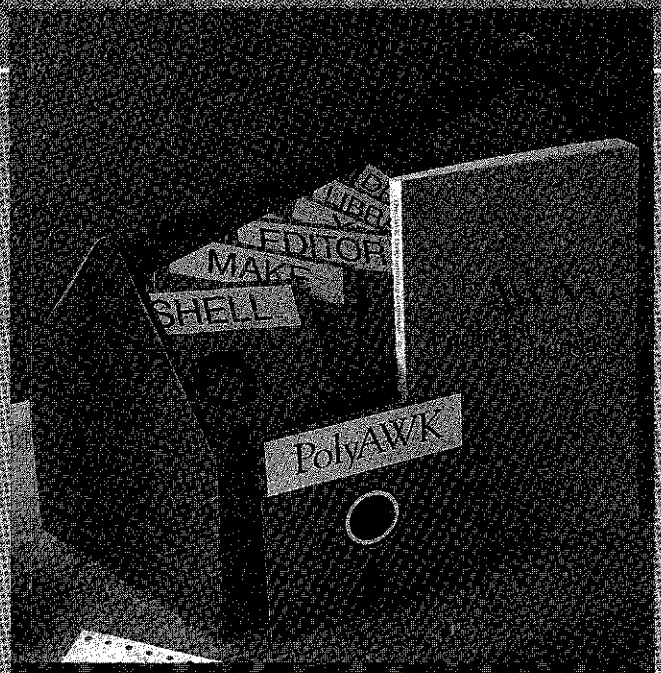
Saves You Time & Effort

The most compelling reason to use PolyAWK is that you can usually accomplish in a few lines of code what may take pages in C, Pascal or Assembler. Programmers spend a lot of time writing code to perform simple, mechanical data manipulation -- changing the format of data, checking its validity, finding items with some property, adding up numbers and printing reports. It is time consuming to have to write a special-purpose program in a standard language like C or Pascal each time such a task comes up. With PolyAWK, you can handle such tasks with very short programs, often only one or two lines long. The brevity of expression and convenience of operations make PolyAWK valuable for prototyping even large-sized programs.

Very Concise Code

Where program development time is more important than run time, AWK is hard to beat. These AWK characteristics let you write short and concise programs:

- The implicit input loop and the pattern-action paradigm simplify and often entirely eliminate control flow.
- Field splitting parses the most common forms of input, while numbers and strings and the coercions between them handle the most common data types.
- Associative arrays use ordinary strings as the index in the array and offer an easy way to implement a single-key database.
- Regular expressions are a uniform notation for describing patterns of text.



PolyAWK.™

The Toolbox Language.™

For C, Pascal, Assembly & BASIC Programmers

OS/2 $199

- Because the interpreter and the absence of limitations shown in programs.

Large Model & Math Support

PolyAWK is a large model implementation and can use all available memory to run big programs or read files greater than 64K. PolyAWK also includes extensive support for math functions such as strings, integers, floating point numbers and transcendental functions (sin, log, etc.) for scientific applications. Conversion between these types is automatic and always optimized for speed without compromising accuracy.

PolyAWK Comes With The Book

When you order PolyAWK you receive a copy of *The AWK Programming Language* written by the authors of the original UNIX-based AWK. The book begins with a tutorial that shows how easy AWK is to use, followed by a comprehensive manual. Because PolyAWK is a complete implementation of AWK as defined by the book's authors, you will use this book as the manual for PolyAWK.

PolyShell Bonus For MS-DOS!

PolyShell gives you 70 of the most useful UNIX commands and utilities under MS-DOS in less than 20K. You can still use MS-DOS commands or any time and even restart PolyShell without rebooting. MS-DOS implementations of what you have been missing if UNIX programmers -- switch to MS-DOS names for PolyShell and PolyAWK, and use $39 when ordered separately. Save $50 by ordering the PolyShell + PolyAWK combination package for $89 (MS-DOS only).

30-Day Money Back Guarantee

You can purchase PolyAWK and the book, *The AWK Programming Language*, for $99 (MS-DOS) or $199 (OS/2), or $249 for both if you already have the book. Add $14 for combination package includes only one book. Order PolyAWK (MS-DOS & OS/2) plus PolyShell (MS-DOS) for $134.

To Order: 1-800-547-4000

Ask for Dept. PB. Send checks and P.O.s to: POLYTRON Corporation, 1700 NW 167th Place, Beaverton, OR 97006. (503) 645-1150. FAX: 503-645-1476. TELEX: 275800 POLYTRON. Call toll-free 24 hours a day, 7 days a week.

POLYTRON

High Quality Software Since 1982

PRODUCED 2014 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Simply the BEST C and Pascal on AT, 386, Sun, Apollo, RT, VAX, 370

"The most rock-solid C compiler in the industry. Superb technical support and portability. Superior code generated."
Gordon Eubanks, Symantec — Q&A (386).

"It simply works, with no trouble, no chasing strange bugs, and excellent warning and error messages ... a professional product."
Robert Lerche, Bay Partners.

"For large-scale software development, the highest quality C compiler available on the market today. Pragmas are great. Quality of support is exceptional." **Randy Neilsen, Ansa—Paradox** (DOS,OS/2).

"15% smaller and 15% faster than Lattice C."
Robert Wenig, Autodesk.

"Our software is running anywhere from 30 to 50% faster than when compiled under Lattice." **David Marcus, Micronetics.**

"We switched from Lattice due to a 10% reduction in code size. The compiler is very stable." **Lee Lorenzen, Ventura Software — Ventura Publisher, marketted by Xerox Corp.**

"Best quality emitted code by any compiler I've encountered. Often amazing." **Bill Ferguson, Fox Software — FoxBase (386).**

"Messages sometimes pointed out type mismatches, incorrect-length argument lists, and uninitialized variables that had been undetected for years [in 4.x bsd]." **Larry Breed, IBM ACIS [RT PC].**

"Diagnostics turned up bugs missed by other compilers. Rapid bug fixes by technical support, someone who knew what he was talking about. 80386 code is well optimized."
Tim Addison, Logistics Data Systems.

"386 protected mode support is fantastic, especially the access to large amounts of memory. It's mainframe compute power on a PC." **Dan Eggleston, Viewlogic.**

"The preprocessor supplied with Professional Pascal is quite useful. The code quality and control over segmentation and memory models are superior to MS Pascal." **Bob Wallace, QuickSoft.**

Check Out These Reviews

- **High C ™:**

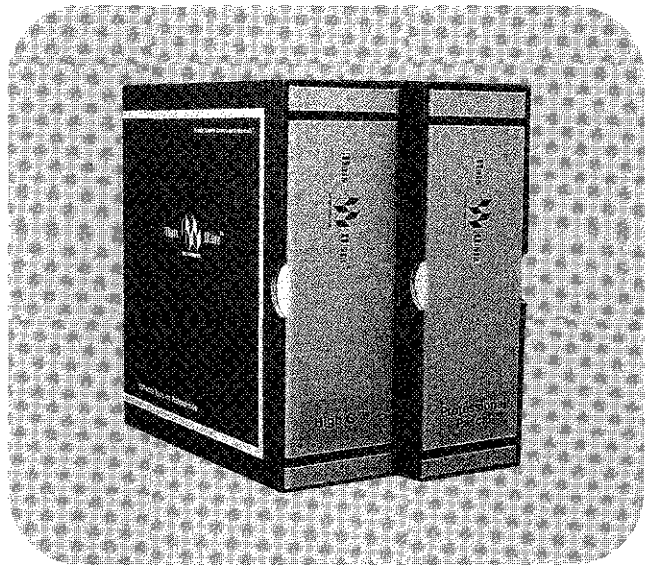
Computer Language	February 1986, '87	
Dr. Dobb's Journal	August 1986	
PC Magazine	Jan. 27, 1987	(80386 version)
Dr. Dobb's Journal	July 1987	(80386 version)
BYTE Magazine	November 1987	(80386 version)

- **Professional Pascal ™:**

PC Magazine	Dec. 29, 1985	
Computer Language	May 1986	
PC Tech Journal	July 1986	
Journal of Pascal, Ada, & Modula-2	Nov.-Dec. 1986	
BYTE Magazine	Dec. '86, June '87	(80386 version)

Why MetaWare compilers

- They are specifically designed for serious software developers.
- They are reliable and robust: they don't break at every turn.
- Their generated code is the best, or near best, on each architecture.
- Their superior diagnostic messages help you produce better products more quickly.
- Your source can be ported with ease to the most popular systems.
- You can link mixed-language modules from our compilers, others
- You can benefit from high-level, personal technical support.
- You can take advantage of the latest ANSI C extensions, and/or extensive Pascal extensions. **High C** has been tracking the ANSI Standard for two years; **Professional Pascal** will soon have a VS Pascal compatibility switch and several Apollo Pascal ext'ns.
- You can take advantage of the latest 387 and Weitek 1167 support — we have the only compilers with Weitek real mode support.



Power Tools for Power Users

Ashton-Tate: dBase III Plus, MultiMate; Autodesk: AUTOCAD, AUTOSKETCH (8087, '387, Weitek); Boeing Computer Services (Sun); CASE Technology (Sun); CAD/CAM giant Daisy ('86, '386, VAX); Deloitte Haskins & Sells; Digital Research: FlexOS; GE; IBM: 4.3/RT, 4680 OS; Lifetree Software (Pascal): Volkswriter Deluxe, GEM-Write; Lugaru: Epsilon; NYU: Ada-Ed cmplr; Semantec: Q&A; Sky Computers; ... (Product names are trademarks of the companies indicated.)

Industrial-Strength

MetaWare C and Pascal compilers are designed for professional software developers. These tools are loaded with options to control them for special purposes. You can adjust the space-time trade-off in code quality. You can adjust external naming conventions to agree with linkers and operating systems. You can specify segment names for segmented architectures, and to help place code or data in particular places for embedded applications. You can select from five memory models for the 8086 family. And on and on.

A Partial List of Optimizations

Common subexpression and dead-code elimination, retention and reuse of register contents, jump-instruction size minimization, tail merging (cross jumping), constant folding, short-circuit evaluation of Boolean expressions, strength reductions, fast procedure calls, automatic mapping of variables to registers (where advantageous), ...

"Platform" — Code Quality

Sun,Apollo,SGI	— 18%, 3%, 26% > resident compiler (Dhrystone).
PC: DOS, OS/2	— 3-10% > Microsoft C; 30% > MS Pascal, LatticeC.
386 32-bit DOS	— no competitors, since November, 1986.
286, 386 UNIX	— 66% better than pcc (Dhrystone, 386).
VAX VMS	— ≈ DEC's excellent C and Pascal; better features.
VAX Ultrix	— 19% > pcc (Dhrystone); much > Berkeley Pascal.
RT PC/4.3bsd	— 89% > the original port of pcc (Dhrystone).
370 CMS,UNIX	— much better than any C, and VS Pascal.
AMD 29000	— >40,000 Dhrystones! Available in Q2, cross.

(408) 429-6382, telex 493-0879. ***Since 1979.***

MetaWare™
INCORPORATED

903 Pacific Avenue, Suite 201 • Santa Cruz, CA 95060-4429

The Clear Choice for Large Programming Projects — PC Tech J.

© 1987 MetaWare Incorporated. MetaWare, High C, Professional Pascal, and DOS Helper are trademarks of MetaWare Incorporated. Others and their owners are duely respected.

Number 45 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal for IBM Micro Programmers

PROGRAMMER'S Journal

Volume 7.1
January/February 1989

Contents

4 Notes from the Editor

7 Index 1988: Issues 6.1 through 6.6

10 Letters
W.G. Madison, Adrian B. Crum, Bill Hall, Alan Weiner

12 Redirections: Pink Noise *by Frank D. Greco*
Techno gossip from our Unix correspondent

18 On Graphics: Higher 256-Color Resolution on the VGA *by Michael Abrash*
When is 320x200 really 320x400, and those less obvious high resolution VGA color modes

33 On Graphics: Migrating to Graphics Processors *by Jake Richter*
The Texas Instruments 34010 and Intel 82786 graphics engines

42 Pascal: Demagnetizing Static Variables *by Tom Swan*
Modifying type constants stored in Turbo Pascal 4.0 and 5.0 compiled EXE code

54 Pascal: My Kingdom for an Overlay! *by Bill Madison*
Comparing Pascal Overlay Managers: TurboPower Software and Borland's Turbo Pascal 5.0

60 Modula-2: From Pascal to Modula-2 *by John J. Newlin*
A review of JPI's TopSpeed Modula-2

64 BASIC: Which Code is Faster? *by Ethan Winer*
Speeding up QuickBASIC string operations, array manipulation, variable assignment, and integer math, along with how to use CodeView with BASIC

72 C: The Proposed ANSI C Language Standard *by Rex Jaeschke*
Rex looks at Lattice C V3.3

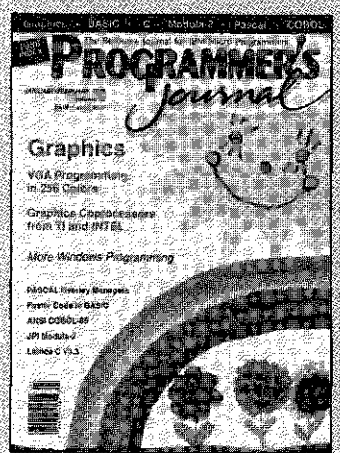
74 C: Smart Terminal Emulation in Windows *by William S. Hall*
A challenging Windows application

84 COBOL: COBOL Comes of Age *by Richard Riehle*
A fresh look at the COBOL-85 ANSI standard

92 Miscellaneous *by M. Steven Baker*
OPUS Make 5.1 (DOS and OS/2)
QuickPak Professional (BASIC utilities)
GOSIP—The government's OSI network standard
Usenix and C++

95 PJ Billboard

96 Advertiser Index



Our preoccupation with color and graphics begins at an early age. The two hundred fifty-six colors of the VGA can bring out the child in anyone.
Cover Artist: Deirdre Bentley, age 6
Art Director: Ralph Bentley

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

SERIOUS DEBUGGING *at a* REASONABLE PRICE

All the speed and power of a hardware-assisted debugger at a software price

Hardware-level break points
REAL-TIME break points on memory locations, memory ranges, execution, I/O ports, hardware and software interrupts. More powerful break points than ANY software-only debugger on the market. Soft-ICE gives you the power of an in-circuit emulator on your desk.

Break out of hung programs
With a keystroke - no external switch necessary. Even with interrupts disabled.

Breaks the 640K barrier
Soft-ICE uses ZERO bytes of memory in the first 1MB of address space. This is especially useful for those subtle bugs that change when the starting address of your code changes. With Soft-ICE your code executes at the same address whether the debugger is loaded or not.

Works with your favorite debugger
Soft-ICE can be used as a stand-alone debugger or it can add its powerful break points to the software debugger you already use. You can continue to use your favorite debugger until you require Soft-ICE. Simply pop up the Soft-ICE window to set powerful real-time break points. When a break point is reached, your debugger will be activated.

Solve tough systems problems too
Soft-ICE is ideal for debugging TSRs, interrupt handlers, self booting programs, DOS loadable device drivers, non-DOS operating systems, and debugging within DOS & BIOS. Soft-ICE is also great for firmware development because Soft-ICE's break points work in ROM.

How Soft-ICE Works
Soft-ICE uses the power of the 80386 to surround your program in a virtual machine. This gives you complete control of the DOS environment, while Soft-ICE runs safely in protected mode. Soft-ICE uses 80386 protected mode features, such as paging, I/O privilege level, and break point registers, to provide real-time hardware-level break points.

"Soft-ICE is a product any MS-DOS developer serious enough to own a 386 machine should have."
Dr. Dobb's Journal — May 1988

RUN CODEVIEW IN ONLY 8K!

CodeView is a great integrated debugger, but it uses over 200K of conventional memory. MagicCV uses advanced features of the 80386 microprocessor to load CodeView and symbols in extended memory. This allows MagicCV to run CodeView using less than 8K of conventional memory on your 80386 PC.

Don't let 640K be your limit!
If you are closing in on the 640K limit and would like the power of CodeView, MagicCV is for you.

Don't let the debugger hide the bug!
Even if you're not closing in on the 640K limit, running CodeView with MagicCV makes your debugging environment much closer to the end user's program environment. You can use CodeView to locate subtle bugs that only occur when there is plenty of free memory, or those difficult bugs that only occur when your program is running with a couple of TSRs loaded.

How MagicCV works
MagicCV uses the 80386 to create a separate virtual machine for CodeView. MagicCV uses between 4K & 8K of conventional memory as a bridge between the DOS environment and CodeView.

MagicCV is easy to use
If you are a CodeView user, you already know how to use MagicCV too. Just type MCV instead of CV; everything else is automatic.

NEW
MS Windows Version

MagicCV	$199
Soft-ICE	$386

Buy Both and Save $86!

CALL TODAY
(603) 888 - 2386
***or FAX* (603) 888 - 2465**

30 day money-back guarantee
Visa, Master Card and AmEx accepted

NU-MEGA TECHNOLOGIES
P.O. BOX 7607 • NASHUA, NH 03060-7607

MagicCV with Soft-ICE
Using Soft-ICE with CodeView gives you the features necessary for professional level systems debugging. MagicCV and Soft-ICE can work in concert with CodeView to provide the most powerful debugging platform you will find anywhere.

Both require 80386 AT compatible or IBM PS/2 Model 80. MagicCV requires at least 384K of extended memory. CodeView is a trademark of Microsoft Corporation.

Number 160 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal For IBM PC Programmers
PROGRAMMER'S Journal

NOTES from the Editor

Editor
M. Steven Baker
Managing Editor
Robert E. Wanetick
Contributing Editors
Michael Abrash, Frank D. Greco, Tom Swan

Contributing Writers
Michael Abrash
M. Steven Baker
Frank D. Greco
William S. Hall
Rex Jaeschke
W.G. Madison
John J. Newlin
Jake Richter
Richard D. Riehle
Tom Swan
Ethan Winer

Publisher
Elizabeth Oakley
Production
Ralph Bentley
Illustration
Jim Carpenter
Advertising
Robert Tappan
Andrea McHone
Subscriptions
Gary Hoyer
Business Manager
Bobbi Sinyard
Administrative Assistant
Lori Fanslow
Founder
Greg Estes

Business and Editorial Offices
Programmer's Journal
150 North 4th Street
Springfield, OR 97477
(503) 747-0800

Subscriptions: 1 year (6 issues) $24
($30 in Canada and $40 elsewhere)
Change of Address
Programmer's Journal
P.O. Box 30160
Eugene, OR 97403

Articles and letters in Programmer's Journal represent the views of their authors and not necessarily those of the publisher. Letters to the editor become the property of Programmer's Journal and may be edited and published.

Code and listings in Programmer's Journal appear "as is," and Programmer's Journal makes no guarantee or warranty of their quality, performance, accuracy, or suitability for any particular purpose. Programmer's Journal assumes no responsibility or liability to any person or entity with respect to any damage caused or alleged to be caused by their use.

Advertising: For ad rates, deadlines, and placement, call Robert Tappan or Andrea McHone (503/747-0800), or write to Programmer's Journal, P. O. Box 30160, Eugene, OR 97403.

Programmer's Journal is produced on Ventura Publisher, a product of Xerox Corporation.

Programmer's Journal (ISSN 0747-5861) is published bimonthly by Oakley Publishing Company. Second-class postage paid at Springfield, OR, and additional mailing offices.

POSTMASTER: Send address changes to Programmer's Journal, P.O. Box 30160, Eugene, OR 97403.

© 1989 *OAKLEY PUBLISHING COMPANY*

Here I sit, the week before Comdex, befuddled by the PC marketplace. I'd rather compose these lines after the show, but the vagaries of magazine schedules intervene. Fall Comdex in Las Vegas has to be experienced for its full impact. The magnitude of the PC industry is rather overwhelming when successful companies can drop $250,000 to $500,000 for a booth at the week-long show. Sure, deals are cut for the coming year; but the importance of publicity and marketing are what Comdex so clearly demonstrates. Be it software or hardware, the best product is unlikely to make it without an enterprising and often costly "dog and pony show."

OS/2 While Technology Waits

Let's slip back eight years in time. In our time capsule we travel to the land of CP/M, those wonderful dual-CPU S-100 boxes from CompuPro, and the funny, little Z-80 portables with teeny screens from Adam Osborne . . . Many small hardware and software vendors without any beanstalk giant . . . Healthy doses of competition and technological advancement.

Enter IBM, from the mainframe world, with their IBM PC . . . Immense success, probably beyond their own expectations. IBM offers standardization, not technological leadership. The marketplace becomes dominated by copies and clones rather than technological advances. But technology can be held back only so long. Now let's return to the present.

IBM and Microsoft have clearly stumbled with OS/2, Presentation Manager, and various extended editions. What at once seemed a "shoo-in," a "piece of cake" for the next operating system and graphics standard, every week slips farther away from reality. If . . . if . . . if DOS 5.0, ADOS, what's now called OS/2, had been delivered two years ago, maybe things would be different now. I attended Comdex last year to see the promised OS/2 software and PS/2 hardware. A year later, not much of substance has materialized. Maybe I'll return next week pleasantly surprised, but I'm not practicing hold my breath. Who knows, maybe I should be studying snorkeling?

Yes, decent OS/2 software development tools are gradually becoming available, but most products are straight ports from the DOS world. User applications are a different story, since they must offer enhanced features such as graphics, sophisticated user interfaces, and greater performance to compete with existing DOS products. That's a big part of the wrinkle. In the past, we complained about PC Unix, since it required 500K for the operating system. Don't ask how much the latest edition of OS/2 with PM takes. Feedback from early business testers is not encouraging. Some suggest 386 machines with 4–8 megabytes of memory as a base configuration for OS/2. While OS/2 slowly evolves into a real product, the 80386 is being stuffed into more and more machines.

But OS/2 doesn't take advantage of any of this 80386 power. The 386 DOS extenders—which by all accounts should never have existed—are alive and well, with new products arriving steadily. Applications based on the 286 DOS extenders from Rational Systems and A.I. Architects are settling into the marketplace. 386 Unix implementations run DOS programs as just another task. With ever-higher-resolution graphics hardware, the likes of X-Window and Display Postscript moving down to the PC are clearly threatening the long-term market for Presentation Manager and its precursor, Windows.

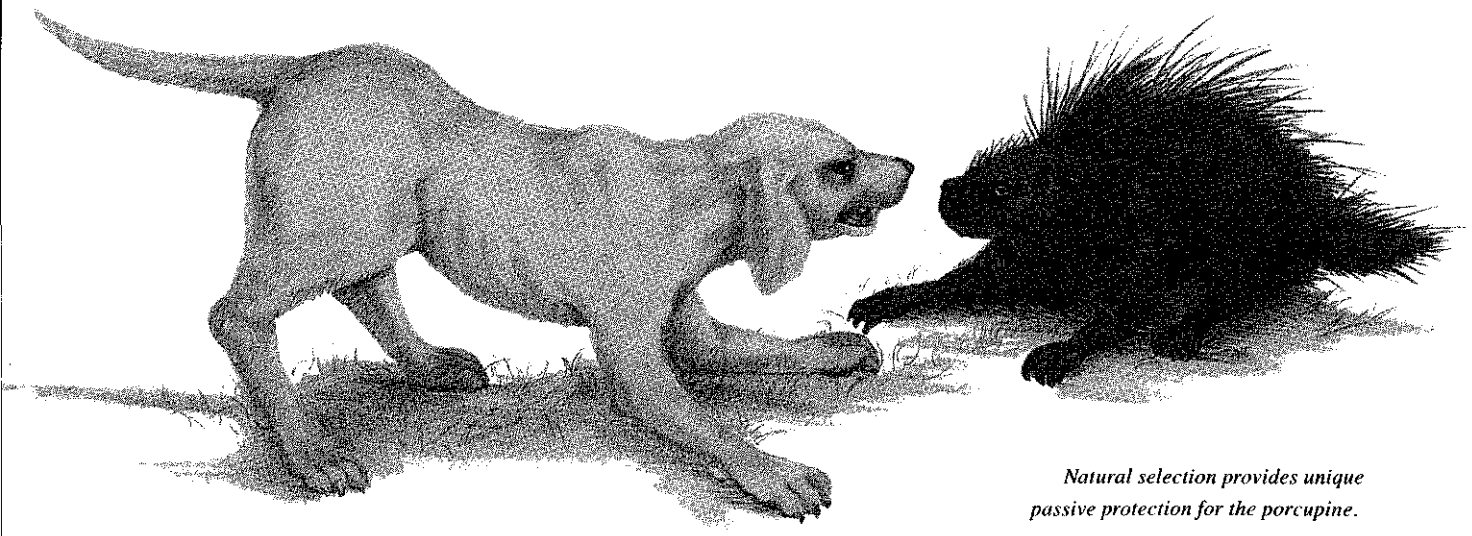
Although IBM and Microsoft have officially met their deadline for delivering Presentation Manager (at least in the press), most developers with OS/2 Software Development Kits are still awaiting the arrival of this latest PM release. I'll wait until the OS/2 developers have a crack at this first "stable" release of PM before holding judgment. Does it really require 400 new system calls for a graphical interface? What ever happened to Sparta?

PJ Staff Changes

A few changes at *PJ*. Bob Wanetick, fresh from editing several popular computer books, has joined our staff as Managing Editor. We welcome him to the growing *PJ* family.

Erin O'Connor, our stalwart editor, has departed for greener pastures. I'm sorry to see her go. She was exceptional—a talented, hardworking, and meticulous journalist who insisted on the highest standards for *PJ*. We wish her well. In her stead, we have a lowly programmer—perish the thought. —*Steve*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Natural selection provides unique passive protection for the porcupine.

The Activator - Natural Selection For Software Protection

Inventor and entrepreneur Dick Erett explains how "The Activator" provides sane protection for your intellectual property.

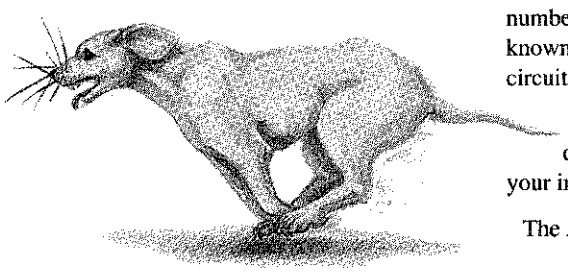
"In any industry, just as in nature, the process of natural selection raises one solution above another. Natural selection is the most elegant of engineers.

In the area of software protection The Block has been selected by the marketplace as the solution that works. Over 500,000 packages are protected by our device.

For the past 4 years our philosophy has been; *'You have the right and obligation to protect your intellectual property.'*

A New Ethic For Software Protection

In allowing end-users unlimited copies of a software package and uninhibited hard disk and LAN operation, The Block has created a new ethic for software protection.



By removing protection from the magnetic media we remove the constraints that have plagued legitimate users.

They simply attach our key to the parallel port and forget it. It is totally transparent, but the software will not run without it.

A New Technology For Software Protection

Our newest model, The Activator, builds on our current patented design, and establishes an unprecedented class of software protection.

We have migrated and enhanced the circuitry of The Block to an ASIC (Application-Specific Integrated Circuit) imbedded in The Activator.

This greatly improves speed and performance, while reducing overall size. Data protection can also be provided.

Programmable Option

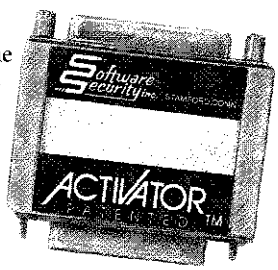
The Activator allows the software **developer** the option to program serial numbers, versions, or other pertinent data known only to the developer, into the circuit, and access it from the program.

Once you program your part of the chip, even we have no way to access your information.

The ASIC makes emulation of the device virtually impossible. It also presents an astronomical number of access combinations.

Full 100% Disclosure

Since The Activator is protected by our patent we fully disclose how it works. Once you understand it, endless methods of protection become evident.



Just as no two snowflakes are the same, no two implementations of The Activator are identical. And like the snowflake the simplicity of The Activator is its greatest beauty.

We never cramp your programming style or ingenuity. Make it as simple or complicated as you desire.

Let us help safeguard what's rightfully yours. Please call today for additional information or a demo unit. *It's only natural to protect your software.*"

1-800-333-0407 ext.103
In Connecticut 203-329-8870
Fax 203-329-7428

Software Security inc.

870 High Ridge Road
Stamford, CT 06905

Unlimited Copies • Programmable • No Batteries • Small Size • Fast • Patented • Data Protection

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C SOURCE for BUSINESS and ACCOUNTING that TURNS LITTLE LEAGUE into GIANTS

What seperates sand-lot from Super League?

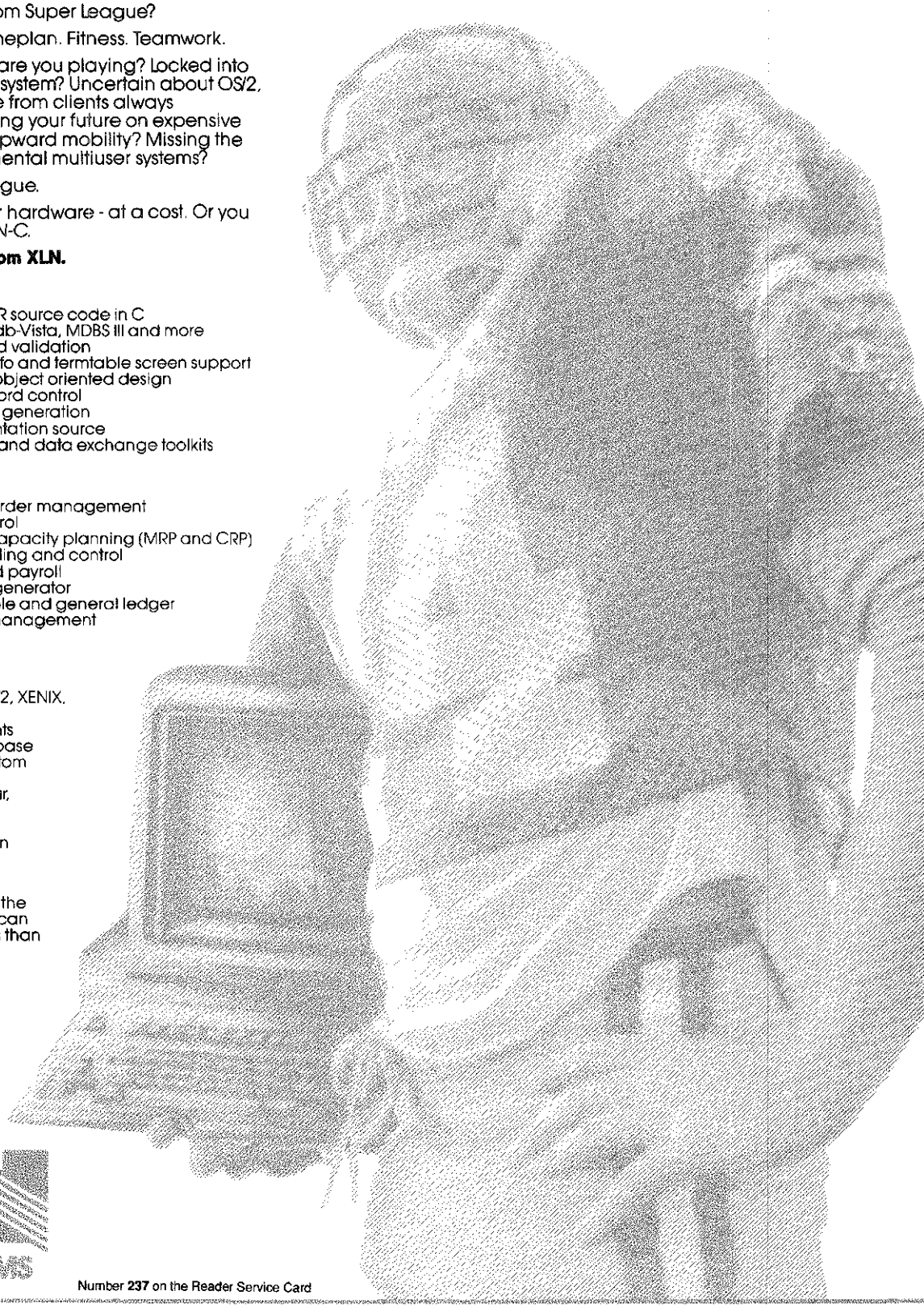
Top Coaches. A better gameplan. Fitness. Teamwork.

So what kind of ball game are you playing? Locked into your proprietory database system? Uncertain about OS/2, UNIX, LAN's? Under pressure from clients always demanding more? Gambling your future on expensive 4GL's? No data sharing or upward mobility? Missing the massive growth in departmental multiuser systems?

That's no way to win the league.

You could standardise your hardware - at a cost. Or you could switch the play to XLN-C.

Buy your C learning curve from XLN.



The skills

- Complete documented K & R source code in C
- Interfaces to c-tree, C-ISAM, db-Vista, MDBS III and more
- Sophisticated data entry and validation
- Video, BIOS, termcap, terminfo and termtable screen support
- Fast, portable and efficient object oriented design
- 1-2-3 style menus with password control
- WYSIWYG screen and report generation
- TEX typesetter and documentation source
- High level graphics, comms and data exchange toolkits
- LISP and Prolog AI system

The plays

- Sales, purchase and works order management
- Multi-location inventory control
- Distribution, materials and capacity planning (MRP and CRP)
- Critical path activity scheduling and control
- Personnel management and payroll
- SQL database application generator
- Accounts receivable, payable and general ledger
- Maintenance and service management
- Fixed asset management

The touchdowns

- No runtime royalties
- Identical source for DOS, OS/2, XENIX, UNIX and VMS
- Portable to 60 + environments
- Upwardly compatible database
- Flexible source code for custom applications
- Grow with standard, modular, integrated object design
- Fast, portable, multiuser, multitasking business solution
- Unlimited technical support

Gate money

$395 gets you in the gates with the module of your choice: or you can cover the whole season for less than the price of a PC.

XLN turns you into a master strategist: while C turns your cowboys into Cowboys.

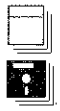
EXCELLENT SYSTEMS

Number **237** on the Reader Service Card

XLN SOFTWARE 3680 Tuggle Road, Buford, GA 30518. Phone orders: 1 (800) 543-5770. Fax: (404) 859-0051. Georgia, International and Support: (404) 945-8745

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

1988: Issues 6.1–6.6



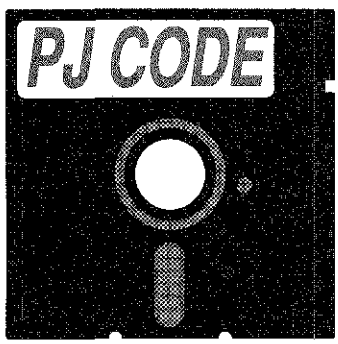
BACK ISSUES and CODE DISKS are available for all 1988 issues. Please see pages 8 and 88 for more information on how to place your order.

Department	Date	Vol.	Page	Author	Title
386	4/88	6.2	64	Smith, Bud E.	80286 and 80386 Programming
386	4/88	6.2	68	Baker, M. Steven	80386 Software Development
386	6/88	6.3	46	Smith, Kevin	Virtualizing the 386
Ada	10/88	6.5	68	Riehle, Richard D.	Ada—A Software Engineering Tool
Ada	12/88	6.6	64	Riehle, Richard D.	Programming for Reliability—Data Types and Exceptions
ASM	4/88	6.2	79	Abrash, Michael	*The Norton Guide for DOS/BIOS Assembly* Reviewed
BASIC	6/88	6.3	68	Winer, Ethan	Comparing CALL, GOSUB and Multi-line Functions
C	2/88	6.1	46	Greco, Frank D.	Time Is on My Side . . . Yes It Is
C	4/88	6.2	47	Jaeschke, Rex	The Proposed ANSI C Language Standard
C	4/88	6.2	50	Young, Michael J.	Converting a Turbo C Program to a TSR
C	8/88	6.4	40	Babcock, Michael & William S. Hall	A Digital Clock for Microsoft Windows
C	8/88	6.4	60	Navas II, John	Windows within Windows
C	8/88	6.4	62	Jaeschke, Rex	The Proposed ANSI C Language Standard
C	10/88	6.5	44	Hall, William S.	An OS/2 Presentation Manager Template
C	10/88	6.5	57	Jaeschke, Rex	Watcom C and the ANSI Standard
C	12/88	6.6	46	Rodgers, Stephen A.	DOS Device Drivers—A Debugging and Information Utility
C	12/88	6.6	53	Jaeschke, Rex	The Proposed ANSI C Language Standard and DeSmet C
C++	10/88	6.5	60	Vernon, Vaughn	Object Oriented OS/2
Consulting	2/88	6.1	12	Barkley, Paul	Consulting Is an Attitude
Consulting	6/88	6.3	30	Barkley, Paul	Getting Your Consulting Tools in Order
Consulting	8/88	6.4	12	Panczenko, Oleg	Job-shopping—Almost Consulting
Consulting	10/88	6.5	16	Barkley, Paul	No-win Jobs and Too Much Work
DOS	12/88	6.6	45	Richter, Jake	64K Extra Memory for DOS Applications
EMS	2/88	6.1	28	Newlin, John J.	The LIM 4.0 Expanded Memory Specification
Forth	12/88	6.6	56	Duncan, Ray	A Forth Apologia
FORTRAN	2/88	6.1	52	Baker, M. Steven	Porting Mainframe Software to the PC
Index	2/88	6.1	71	PJ Staff	Five-year Index of PJ Articles
Misc.	2/88	6.1	76	Baker, M. Steven	80386 Development Tools
Misc.	2/88	6.1	78	Baker, M. Steven	386 Compilers
Misc.	2/88	6.1	80	Baker, M. Steven	Windowing and Graphics
Misc.	2/88	6.1	80	Baker, M. Steven	Standards (FORTRAN 8X)
Misc.	2/88	6.1	80	Baker, M. Steven	Books (*Mastering Turbo Pascal Files*) by Tom Swan
Misc.	4/88	6.2	80	Baker, M. Steven	Program Translators
Misc.	4/88	6.2	82	Baker, M. Steven	Communications (Portal System, DDN Protocols)
Misc.	4/88	6.2	82	Baker, M. Steven	Standards (P1003.1 POSIX)
Misc.	4/88	6.2	82	Baker, M. Steven	Free Software Foundation
Misc.	4/88	6.2	84	Baker, M. Steven	Books (*The MS-DOS Encyclopedia*)
Misc.	6/88	6.3	71	Baker, M. Steven	80286/80386 Stuff—LMI UR-, OS/2- and TMS34010-FORTH, Lahey FORTRAN F77L-EM, Soft-ICE 1.0
Misc.	6/88	6.3	74	Baker, M. Steven	Graphics/windowing—Windows SDK, TMS 34010 Development Tools, Logitech HIREZ Mouse, DEMO II
Misc.	6/88	6.3	76	Baker, M. Steven	Parser Generators—LALR 3.0, QPARSER+ 3.0
Misc.	6/88	6.3	77	Baker, M. Steven	Assemblers/Tools—OPTASM, C Programmers Toolbox
Misc.	6/88	6.3	78	Baker, M. Steven	Books—*Programmer's Guide to PC and PS/2 Video Systems*, Advanced C Programming for Displays
Misc.	8/88	6.4	77	Baker, M. Steven	Windowing and Graphics—*Microsoft Mouse Programmer's* Reference, GEM/3 Programmer's Toolbox, Prospero GEM Pascal & FORTRAN
Misc.	8/88	6.4	77	Baker, M. Steven	Standards—System V Interface Definition
Misc.	8/88	6.4	78	Baker, M. Steven	C Math Libraries—C Scientific Library, C Math Toolbox, Vol.I
Misc.	8/88	6.4	79	Baker, M. Steven	Compilers and Languages—NDP C-386 Compiler, Watcom C6.0, Guidelines C++ V1.2, OS/2 & DOS C5.1, Pascal 4.0, FORTRAN 4.1, & BASIC 6.0 compilers, MASM 5.1 & OS/2 Toolkit
Misc.	10/88	6.5	87	Baker, M. Steven	Program Editors and Tools—MKS Toolkit V2.3, BRIEF 2.1, PI Edit V2.0
Misc.	10/88	6.5	90	Baker, M. Steven	Debuggers—Periscope V4, MagicCV
Misc.	10/88	6.5	92	Baker, M. Steven	Translators—FOR_C
Misc.	12/88	6.6	90	Baker, M. Steven	Make Utilities—PolyMake, Seidl Make and SMKgen, Text Editors and Formatters—MKS SoftQuad Publishing Software, SEE Editor
Misc.	12/88	6.6	94	Baker, M. Steven	C:LINES/C:TREE
Misc.	12/88	6.6	94	Baker, M. Steven	Multitasking—DESQview 2.0 API Tools

continued . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Department	Date	Vol.	Page	Author	Title
Numerics	12/88	6.6	72	Baker, M. Steven	Crunching Numbers at Blazing Speed
OS	6/88	6.3	38	Letwin, Gordon	OS/2 Memory Management from *Inside OS/2*
On Graphics	2/88	6.1	16	Abrash, Michael	Write Mode 3 of the VGA
On Graphics	4/88	6.2	26	Abrash, Michael	Yet Another VGA Write Mode
On Graphics	8/88	6.4	22	Abrash, Michael	Reading VGA Memory
On Graphics	8/88	6.4	32	Baker, M. Steven	X Window—A View from DOS
On Graphics	10/88	6.5	22	Abrash, Michael	VGA Text Paging
On Graphics	12/88	6.6	32	Abrash, Michael	Saving VGA Screens, Color Selection, and Other Mysteries
Pascal	2/88	6.1	38	Swan, Tom	Routines to Access the Environment (TP4)
Pascal	4/88	6.2	38	Newlin, John J.	Inside Turbo Pascal 4.0
Pascal	6/88	6.3	56	Swan, Tom	A CASE of the Jumps
Pascal	6/88	6.3	60	Newlin, John J.	Turbo Professional 4.0 reviewed
Pascal	8/88	6.4	66	Newlin, John J.	Not a Creature Was Stirring—A Mouse Unit for Turbo Pascal 4.0
Pascal	10/88	6.5	34	Swan, Tom	Programming on the Variant Express
PJ Forum	4/88	6.2	14	Hardenbergh, Hal W.	RISCs, CISCs, and Fabs
PJ Forum	6/88	6.3	16	Hardenbergh, Hal W.	Memory Trends
PJ Forum	10/88	6.5	12	Hardenbergh, Hal W.	Computers in the Fast Lane
PJ Forum	12/88	6.6	19	Madison, William G.	DOS 4.0 Leaps into the OS Mix
Redirections	4/88	6.2	20	Greco, Frank D.	Pink Noise
Redirections	6/88	6.3	20	Greco, Frank D.	Pink Noise
Redirections	8/88	6.4	18	Greco, Frank D.	Windowing Systems Overview—MS Windows/Presentation Manager, GEM, X Window, Sun NeWS, Open Look, the Apple Macintosh
Redirections	12/88	6.6	22	Greco, Frank D.	OS/2 vs. Unix
REXX	10/88	6.5	78	Chalk, Andrew J.	The REXX Language on the PC



Git yer *PJ Code DiskScription!*

HOT CODE—
COOL PRICES—WE DELIVER!

Typing in long code listings uses up a lot of valuable time, and calling up bulletin boards doesn't appeal to everyone. So what does the modern programmer do? Get a ***PJ Code DiskScription***, of course!

What's a PJ Code DiskScription? All the code listings from six consecutive issues of *Programmer's Journal*, sent directly to you every other month on disk, for the money-saving subscription price of only $39.95.*

You'll get all the upcoming installments of Michael Abrash's EGA/VGA ASM code, more of Tom Swan's handy Pascal code, plus additional great code from more of the country's hottest programmers.

You'll receive your code disks in the mail every other month throughout the year.

Learn from ***PJ*** code or use in it your own programs. Each disk contains all of the program listings published in ***PJ*** plus .EXE files, .OBJ files, benchmark source files, and more!

If purchased separately, the regular price for these 6 disks is $60.00. **So save yourself time, effort and over $20.00—order your *PJ DiskScription* today!**

Programmer's Journal PJ Code DiskScription—
Code disks from 6 issues for only

$39.95*

* **Prepaid, US only.** Call or write for prices outside the US.

GUARANTEE: *If you are dissatisfied with your* ***PJ Code DiskScription*** *for any reason, we will gladly send you a refund on all unmailed diskettes.*

GET HOT *PJ* CODE DELIVERED DIRECT TO YOU.
START YOUR *PJ Code DiskScription* TODAY!

PJ Code DiskScriptions must be prepaid. Return the card on the opposite page with your payment.
For faster service call toll-free: 800-234-0386 with your VISA or MASTER CARD.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

THE PROGRAMMER'S SHOP

helps save time, money, and cut frustrations. Compare, evaluate, and find products.

NEW Catalog!

Our new, colorful, completely updated catalog lists over 700 of the most popular, cutting edge programmer's tools, clearly organized into intuitive function categories. Completely indexed by product and manufacturer plus helpful notes on every item listed, common questions and answers. Yours FREE when you call.

New Discoveries

Microsoft Programmer's Library
48 references on CD-ROM disk include Windows and OS/2 development kit manuals, Microsoft Press books, most Microsoft language manuals, and 20 floppies' worth of indexed sample code. Indexed and cross-referenced. $329, Additional network nodes $59.

Turbo Assembler/Turbo Debugger
Borland MASM-compatible, nested structures, unions; x-ref multiple modules, assemble multiple files; type-checking. 386 support, use with Turbo C, Pascal, others. Fast, with extensions. Debugger has multiple overlapping windows, 386 virtual-86 debug; remote, data debug; conditional breakpoints. CodeView-compatible support. $99

C86PLUS/ROM Development System
Computer Innovations' ANSI C embedded systems environment with symbolic debug, memory-control (cuts ROM-RAM startup data copy, enhanced for 80x/86), startup code for non-DOS, ANSI runtime library (trig, floating point) with source, linker/locater. No royalties. $1159

386 Development Tools

	List	Ours
386 Assembler/Linker	495	399
FoxBASE+/386	595	399
High C - 386	895	Call
Hoops/386-32	575	489
Lahey Fortran F77L-EM/32	895	Call
NDP C-386	595	539
NDP Fortran-386	595	539
Paradox/386	895	639

AI-Expert Systems

CxPERT V3.0 - shell for C	395	389
Exsys Professional	795	695
Personal Consultant Plus	2950	2589

AI-Languages

A.P.T. - Active Prolog Tutor	65	49
ARITY Combination Package	1095	979
PC Scheme LISP - by TI	95	79
TransLISP PLUS - call MSC	195	119
TURBO PROLOG V2.0	150	109
Turbo Prolog Toolbox	100	69

Assemblers

MS Macro Asm	150	105
OPTASM - 75K lines/minute	125	109
Turbo Assembler/Debugger	150	105
Visible Computer 80286	100	89

BASIC & Addons

BAS_C com. - BASIC to C	375	329
Exim Toolkit	100	85
LaserPak Professional	149	119
MS QuickBASIC V4.0	99	69
QuickPak Professional	149	119

C Language - Compilers

AZTEC C86 - Commercial	499	Call
High C-286 - by MetaWare	595	Call
Lattice C - V3.3	450	289
Microsoft C 5.1 - w/CodeView	450	299
Microsoft QuickC	99	69
Turbo C - by Borland	150	105
Watcom C6.5 - highly optimized	295	269

C Language - Interactive

C-terp - by Gimpel, K&R	300	229
C Trainer - by Catalytix	124	109
Instant-C - source debug	495	359

CASE

Dan Bricklin Demo II	195	179
Interactive EasyFlow	150	119
Matrix Layout - flow chart	150	139
Show Partner F/X - demos	350	299

COBOL

MS COBOL V3.0 - OS/2	900	539
Realia COBOL	995	849

Communications Addons

C Asynch Manager - Blaise	175	135
Essential Comm Library	185	129
with Breakout Debugger	310	209
Greenleaf Comm Library	229	169

DataBase Mgmt

Advanced Revelation	950	779
CLARION - complete envt.	695	589
DataFlex - by Data Access	695	599
Eagle - by Migent	495	419
Magic PC - visual database	199	179
Paradox V2.0 List: $725	725	529
R:base for DOS	725	539
Sycero	695	669

Dbase

	List	Ours
Clipper Summer '87	695	449
dBASE IV	795	529
DBXL Interpreter	199	149
FoxBASE+ - V2.1	395	249
QuickSilver Diamond	599	369

Dbase Addons

dBASE Tools for C	90	75
dGE - business graphics	195	179
FLIPPER Graphics Library	195	169
IDL Integrated Devt Library	149	149
Seek-It - query by example	100	89
SilverComm Library	150	129
Tom Rettig's Library	100	79

Dbase Tools

CLEAR+ for dBase	200	159
dBRIEF w/BRIEF	275	Call
dBug - source debugger	195	179
dBX - dBASE III to C	550	529
Documentor	295	239
FoxTool Box	295	279
Genifer - code generator	395	259
R&R Relational Report	149	139
R&R Clipper & FoxBASE	55	45
UI Programmer	295	249

Debuggers

Periscope II - breakout switch	175	139
Periscope III - 10 MHz version	1395	1129
Pfix-86 Plus	395	219

Development Tools

Instant Replay	150	139
Lattice Compiler Companion	100	Call
MKS AWK	99	89
PC - Lint	139	109
PC - Metric - analyze	199	189
Plink 86 Plus - overlays	495	299
PolyMake	149	135
PVCS Corporate	395	369
Seidl Version Manager	300	279

Editors

BRIEF	195	Call
Edix	195	159
Epsilon - like EMACS	195	149
KEDIT - like XEDIT, V.4	150	125
Personal Rexx	125	109
ME Macro Editor - Source	189	159

Editors Cont.

	List	Ours
MKS Vi	75	66
EDT+ 4.0 - VAX EDT on PC	295	269
SPF/PC - V2.0	245	189
Vedit Plus	185	135

File Addons

Btrieve ISAM	245	185
Btrieve/N - multiuser	595	455
Report Option	145	119
XQL - SQL for Btrieve	795	599
Xtrieve	245	189
c-tree by Faircom - source	395	319
d-tree - data dictionary	495	419
r-tree - report generator	295	239
c-tree - w/r-tree	650	519
CBTREE - Source	159	129
dBC ISAM III - by Lattice	250	169
dB_FILE - network model	varies	Call
dB_RETRIEVE - SQL	varies	Call
Opt Tech Sort	149	99
WKS Library	195	179

FORTRAN

FOR_C - FORTRAN '77 to C	750	679
I/O Pro - screens	149	129
Lahey FORTRAN F77L	477	459
Lahey Personal FORTRAN	95	89
MS FORTRAN - CodeView	450	299
RM/FORTRAN	750	479
Scientific Subroutines	175	125
Spindrift - DOS, windows	149	129

General Addons

C Tools Plus - V5.0	129	99
C Utilities - by Essential	185	129
Greenleaf SuperFunctions	265	209

Graphic Addons

Drawbridge	129	109
Essential Graphics	299	229
with source	598	519
GraphiC - like DISSPLA	395	309
GSS Dev't Toolkit	495	409
Halo '88 - 140+ devices	325	229
Hoops 3D Graphics	575	549
MetaWINDOW/PLUS	275	219
PCX Publishing's Toolkit	90	85
QuickWindows/C	95	79
Turbo Plus	100	95

Macintosh

	List	Ours
FoxBASE+ Dev't Package	395	339
McMax by Nantucket	295	269
Lightspeed C	175	159

Call for more Mac info

Obj.-Oriented

Actor	495	419
C-Talk	150	129
Smalltalk/V	100	85
Smalltalk/V 286	200	169
Zortech C++ - true compiler	100	99

OS - Support

DESQview	130	115
MS Windows/286	99	69
MS Windows/386	195	139
Development Toolkit	500	329
MKS Toolkit - Unix shell	199	179
OS/2 Programmer's T'lkit	350	249
PC/MOS 386 5 User	595	529
Timeslicer	295	265
VM/386 Multitasker	245	229
XTree Pro	129	109

Other Languages

APL Plus - by STSC	695	549
CCSM Mumps - single	160	149
Modula-2 Dev. System	249	209
RPG II complete	1400	1299
SNOBOL4+ - strings	95	85

Other Products

AEWindows	150	129
Back-It	130	119
CO/SESSION	249	209
Crosstalk XVI	195	129
Disk Technician Plus	130	119
Link + Locate++ - ROM MSC	395	349
Mace Utilities	99	85
Norton Utilities Advanced	150	109
PC Tools Deluxe	80	75
SoftProbe II/TX - debug	395	345
SPSS/PC Plus	795	749
TK Solver Plus	395	339
Watchdog - security	295	265
WordPerfect	495	299
Zap - file xfer	95	89

Text Screen Addons

C Worthy w/forms	295	Call
with source	495	Call
Curses - by Aspen Scientific	119	109
Greenleaf DataWindows	295	229
PowerScreens - by Blaise	129	99
Vitamin C - source, menus	225	169
VC Screen - painter	149	125
Windows for Data	295	265
with source, OS/2 or Xenix	590	Call

Unix/Xenix

DosMerge 286	249	219
DosMerge 386	499	479
Edix - editor	275	249
JAM JYACC Applic. Mgr.	many	Call
Microport System V/286	898	795
Microport System V/386	1198	999
SCO Xenix System V 286	1295	975
SCO Xenix System V 386	1595	1439
Windows for Data	795	719

Note: Mention this ad. Some prices are specials. Ask about COD and POs. Formats 3" laptop now available, plus 200 others. UPS surface shipping add $3/per normal item. All prices subject to change without notice.

Established 1983. We carry over 1,100 *different* products. Call today for complete technical information and advice, our catalog, literature, and *solid value*.

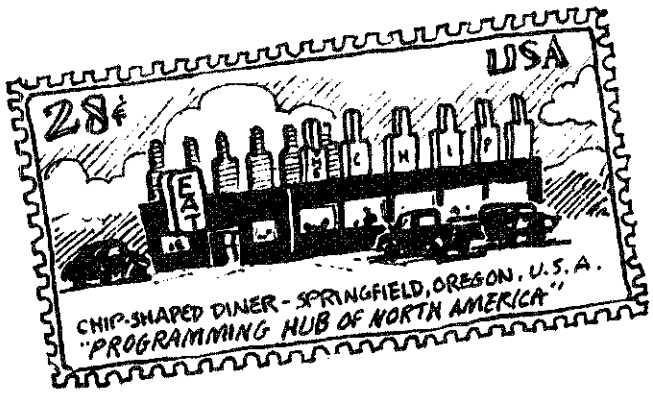
5-J Pond Park Road
Hingham, MA 02043
Mass: 800-442-8070
or 617-740-2510 11/88
Telex: 671-5348
FAX: 749-2018

800-421-8006

HOURS:
M-F 8:30 - 8:00
Sat: 10:00 - 4:00

Number **114** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Here We Go Again
(And It's About Time)

I was delighted to see Terry Ritter's response to my note about structured programming (*PJ* 6.5). Here is clearly someone who refuses to swallow the whole cow regardless of how appetizingly it's presented!

Mr. Ritter is in good company. Several years ago, one of the computing luminaries (Don Knuth, I think) published an article entitled "Structured Programming with GoTo's." At about the same time, someone else (I forget who) published an article entitled "Structured Programming—Good Heuristics, or Bad Dogma?"

The titles say it all. There is nothing inherent in the GoTo that *forbids* construction of clear, understandable, maintainable code. Nor is there anything magical about structured programming that *guarantees* that the resulting code will be of stellar quality.

This misses the real point of Dijkstra's letter, however. Dijkstra's real contention was that the presence and undisciplined use to the GoTo precluded any algorithmic proof of correctness. The fact that non-use of the GoTo generally (but not always, as correctly pointed out by Mr. Ritter) results in more readable code is serendipity. Nothing more.

Regarding Mr. Ritter's comments that a computer program is a form of literature, one cannot reasonably disagree. But the concept needs to be carried even further, into the realm of program documentation.

My view here completely agrees with that of Dijkstra:

"Eventually, there are two 'machines.' On the one hand there is the physical machine that is installed in the computer room. . . . On the other hand there is the abstract machine as defined in the manual, the 'thinkable' machine for which the programmer programs and with respect to which the question of program correctness is settled.

"Originally I viewed it as the function of the abstract machine to provide a truthful picture of the physical reality. Later, however, I learned to consider the abstract machine as the 'true' one, because that is the only one we can 'think'; it is the physical machine's purpose to supply a (hopefully) sufficiently accurate physical simulation of the true, abstract machine." (Dijkstra, E.W., *A Discipline of Programming*, Addison-Wesley, 1976, p. 201.)

Replace the word *machine* with *program* and *programmer* with *user*, and make a few other necessary editorial changes. Now try to justify changing the documentation to make it fit the program instead of vice versa.

W.G. Madison
Greenbelt, MD

Apologies to Swan

Please allow me to apologize to Tom Swan and the *PJ* readers for any confusion that I may have caused with my letter in *PJ* 6.4. When I read "A Case of the Jumps," I thought that Mr. Swan was simply trying to find another way to code a Case statement. I felt that my suggestion would run faster than the TP4 cascaded CMP, JMP statements. What I meant by "more efficient" was better use of memory; I was aware that Mr. Swan's code would run faster. Since I learned to program in the CPM days, I balked at the wasted memory.

Michael Frame's "compromise code" is a very good idea. It allows fast execution time and memory savings. For what it's worth, here is the assembly code:

```
CODETABLE DB     49 DUP(0),1,2
          DB     37 DUP(0),3,40 DUP(0)
JTABLE    DW     ENDC,CASE1,CASE2,CASEX
          MOV    BX,OFFSET CODETABLE
          AND    AX,7FH
          XLAT
          SHL    AX,1
          MOV    BX,AX
          JMP    JTABLE[BX]
          ...
          ...
CASE1:
CASE2:
CASEX:
ENDC:
```

In this example, valid selectors are 1, 2, and X.

Adrian B. Crum
La Habra, CA

OS/2 Presentation Manager Changes

Since my article "An OS/2 Presentation Manager Application Template" appeared in *PJ* 6.5 (September/October 1988), Microsoft has altered the API enough to break the original code. In particular, WinCreateStdWindow has changed. I am including a new version, which should compile and run correctly under the current PM SDK, version 4.99. I hope you can make the new code available on the *PJ* Bulletin Board.

Bill Hall
La Habra, CA

[Thanks, Bill, for keeping us current.—Ed.]

The Weiner Shell Language

In "The REXX Language on the PC" (September/October 1988, Volume 6.5), Andrew Chalk points out that although REXX does extend the DOS batch language somewhat, it has many limitations. Some of those limitations are 384K of memory overhead, minimal DOS-specific features, and no interrupt capability. I'd like to let your readers know about a product that addresses all these limitations. The Weiner Shell language from GRYPHON Microproducts adds about 140 commands to DOS and to any other software being run. Its features include array-processing, floating-point math, pop-up scrollable windows, light-bar menus, user I/O control, table look-ups, background telecommunications, DOS interrupts, and database (dBASE) file access. The language is extendable, which means that users can define their own functions, even those written in other languages, and add them to the Shell. The Shell engine is memory-resident. It requires as little as 45K and can take advantage of LIM-spec memory. It may be removed from memory without rebooting the system. Business-Graphics and Statistics add-ons are also available.

The Shell is a high-level language, similar to a cross between dBASE and Assembler. All commands are executed by "pushing" parameters onto a stack and then executing the stack. Each parameter is surrounded by a delimiter. The default delimiter is the char ~ and may be changed by the programmer. Each command line begins with a command verb and is followed by its parameter list. For example, to print the string "Hi y'all world" to the screen, the following syntactical elements would be used:

```
~atsay~ ;This is the verb
~Hi y'all world~ ;this is the first
    (only) parameter
~~ ;a null parameter says "run the stack"
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Placing this all together, the command line would read:

```
~atsay~~Hi y'all world~~~
```

In Shell terminology, this is called a delimited string. The actual line from a DOS batch file would be

```
NULL ~atsay~~Hi y'all world~~~
```

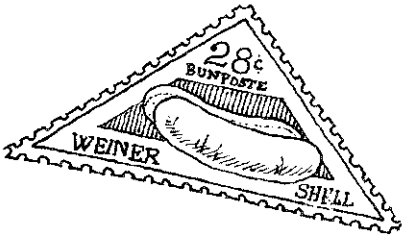
The Weiner Shell also has a more natural (squiggle-less) syntax for DOS.

Results of an operation are communicated to the calling program in several programmer-defined ways: through the DOS environment, through the ICA, through the keyboard buffer, through Shell system variables, through arrays, through the serial port, through the printer port, or through use of files. In addition to being able to call all 140 commands directly from DOS, you can call commands from other applications and languages. There are seven different ways to call programs written in the Shell language: (1) direct memory calls to the Shell interrupt (usually 7Fh); (2) timed events; (3) screen events; (4) activated by hot-keys; (5) started when specified background communications events occur; (6) activated by printing delimited strings to the Printer; and (7) activated by printing delimited strings to the NUL device.

All the Shell commands are based on arrays. Arrays may be created and erased dynamically. They can be saved to and restored from disk, in both compressed and ASCII-text form. Normally, an array contains data, but in the Shell language, an array may also contain screen snapshots and programs. Programs can be sequences of Shell commands (command arrays) or user defined functions (UDFs) written in other languages. About 120 of the Shell commands are actually UDFs. Any of those UDF arrays may be erased to lower the memory requirements of the system. Arrays created in one environment (in LOTUS, for example) are maintained after that application terminates. Arrays may therefore be used as a pipeline to cut and paste data between applications.

Additional information is available on our bulletin-board at (301) 384-7238.

Alan Weiner
GRYPHON Microproducts
Silver Spring, MD ♦



Unbelievable!

SOURCER™

■ **SEE HOW PROGRAMS WORK**
■ **EASILY MODIFY PROGRAMS**

SOURCER™ creates detailed commented source code and listings from memory and executable files. Built in data analyzer and simulator resolves data across multiple segments and provides detailed comments on interrupts and subfunctions, I/O ports and much more. Determines necessary assembler directives for reassembly. Includes a definition file facility to include your own remarks and descriptive labels, force data types, and more. Complete support for 8088/87 through 80286/287 and V20/V30 instruction sets. We welcome comparisons with any other product, because no product comes close to the ease of use and output clarity of SOURCER.

On my list of programs that I simply won't do without!
—Robert Hummel, Senior Technical Editor, PC Magazine

SAMPLE OUTPUT

Fully automatic
Program header
Assembler directives
Determines data areas and type
Detailed comments
Simulator follows segment changes
Easy to read format

```
resetprn.lst    ResetPRN v1.01              Sourcer Listing    19-Apr-88  4:05 pm     Page 1

                                PAGE  60,132
                                ;-----------------------------------------------------------------------
                                ;                              RESETPRN
                                ;
                                ;       Created:   15-Apr-88
                                ;       Version:   1.01
                                ;-----------------------------------------------------------------------

= 0008                          data_1e         equ     8               ; (0040:0008=378h)

                                ;---------------------------------------------------------- seg_a -----
                                seg_a           segment para public
                                                assume cs:seg_a, ds:seg_a, ss:stack_seg_b

                                resetprn        proc    far
658E:0000                       start:
658E:0000  EB 23                                jmp     short loc_1
658E:0002  52 65 73 65 74 50                    db      'ResetPRN v1.01', 0Dh
658E:0008  52 4E 20 76 31 2E
658E:000E  30 31 0D
658E:0011  0040                 data_2          dw      40h
658E:0013  0D 0A 52 65 73 65    data_3          db      0Dh, 0Ah, 'Reset Printer? $'
658E:0019  74 20 50 72 69 6E
658E:001F  74 65 72 3F 20 24
658E:0025                       loc_1:
658E:0025  0E                                   push    cs
658E:0026  1F                                   pop     ds
658E:0027  BA 0013                              mov     dx,offset data_3  ; (658E:0013=0Dh)
658E:002A  B4 09                                mov     ah,9
658E:002C  CD 21                                int     21h               ; DOS Services  ah=function 09h
                                                                          ;  display char string at ds:dx
658E:002E  B4 01                                mov     ah,1
658E:0030  CD 21                                int     21h               ; DOS Services  ah=function 01h
                                                                          ;  get keybd char al, with echo
658E:0032  3C 79                                cmp     al,79h            ; 'y'
658E:0034  75 16                                jne     loc_3             ; Jump if not equal
658E:0036  8E 1E 0011                           mov     ds,data_2         ; (658E:0011=40h)
658E:003A  8B 16 0008                           mov     dx,ds:data_1e     ; (0040:0008=378h)
658E:003E  83 C2 02                             add     dx,2
658E:0041  B0 08                                mov     al,8
658E:0043  EE                                   out     dx,al             ; port 37Ah, printer-2 control
                                                                          ;  al = 8, initialize printer
658E:0044  B9 8000                              mov     cx,8000h
658E:0047                       locloop_2:
658E:0047  E2 FE                                loop    locloop_2         ; Loop if cx > 0
658E:0049  B0 0C                                mov     al,0Ch
658E:004B  EE                                   out     dx,al             ; port 37Ah, printer-2 control
                                                                          ;  al = 0Ch, init & strobe off
658E:004C                       loc_3:
658E:004C  B4 4C                                mov     ah,4Ch            ; 'L'
658E:004E  CD 21                                int     21h               ; DOS Services  ah=function 4Ch
                                                                          ;  terminate with al=return code
                                resetprn        endp
                                seg_a           ends

                                ;---------------------------------------------------------- stack_seg_b ----
                                stack_seg_b     segment para stack
6593:0000  00C0[FF]                             db      192 dup (0FFh)
                                stack_seg_b     ends

                                                end     start
```

(Source code output and inline cross reference can also be selected)

BIOS SOURCE

■ **CHANGE AND ADD FEATURES**
■ **CLARIFY INTERFACES**

for PS/2, AT, XT, PC, and Clones

The BIOS Pre-Processor™ with SOURCER provides the first means to obtain accurate legal source listings for any BIOS! Identifies entry points with full explanations. Resolves PS/2's multiple jumps for improved clarity. Provides highly descriptive labels such as "video_mode" and much more. Fully automatic.

SOURCER $99.95 BIOS Pre-Processor* $49.95 SOURCER w/BIOS Pre-Processor $139.95

USA Shipping & Handling $3; Outside USA $15; CA Residents add local sales tax 6, 6.5 or 7%; *requires SOURCER

All our products come with a 30 day money back satisfaction guarantee. Not copy protected. To order or receive additional information just call!

VISA **(800) 662-8266** MasterCard

V COMMUNICATIONS

3031 Tisch Way, Suite 905, Dept. J3, San Jose, CA 95128 (408) 296-4224

PS/2, AT, XT, and PC are trademarks of IBM Corp.

Number **125** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Pink Noise

by
Frank D. Greco

X.12?
One of the X Window System's several technical drawbacks is that it's a very resolution-dependent (i.e., pixel-based) window system. Sun Microsystems' elegant NeWS window system, based on PostScript, is totally independent of screen resolution. The scuttlebutt among certain people at the last Usenix convention: A good bet that the next generation X Window System, X.12, will be very similar to NeWS today. Although technically inferior to NeWS, the X Window System probably will be very popular in the near-term future (two to four years). In the long run, however, the sheer elegance of NeWS will win out—*if* Sun applies some *significant* effort toward fixing the bugs and enhancing the package. If Sun—or some third party—does not produce some significant software product using NeWS, then we can expect X Window to survive, leaving NeWS to be a technically interesting but commercially unsuccessful product. I certainly hope that we don't have a repeat of the Sony Betamax versus VHS battle. (If you recall, Sony's Betamax VCR technology was technically superior, but the industry-standard VHS proved victorious.)

Rooms for Rent?
A recent issue of PC Week had Xerox-spinoff Envos talking to Microsoft about Envos's "Rooms" graphical metaphor. I wonder if Microsoft knows about the licensing arrangement between Xerox and Sun regarding user interface technology?

Ring Ring at Sing Sing
Did you know that prisons are a major source of income for AT&T? It seems that prisoners call their relatives and friends and remain on the phone all day long.

I Object
The originator of the Ingres database, Dr. Michael Stonebraker, has been busy at work implementing his newest invention, Postgres. Postgres is an object-oriented layer around a conventional relational database. While there is some benefit to basing Postgres on the relational model—backward compatibility with a query language for one thing—I really feel that we must take a step backward and build object-oriented databases from scratch. It just does not "feel" right to build an object-oriented database out of a relational one; it's really a kludge. In my opinion, the number of relations required to formulate an arbitrarily complex, object-based query will increase so much as to significantly degrade performance. Anyway, rumor has it that Stonebraker's boys at Berkeley are also attempting to put Postgres on a chip.

Someone in a California university has been busy at work reverse-engineering the Adobe font metrics (the so-called "proprietary hints"). Within six months, PostScript clones will all be able to license this work.

With the acquisition of Folio, Sun Microsystems is actively incorporating Folio's sophisticated font technology into NeWS. By doing so, the PostScript-based NeWS will be able to use non-Adobe fonts in any point size. Folio's software also handles generating the so-called "hints" to render fonts aesthetic (and readable) at small point sizes.

A RISC for Apple
Very strong evidence exists for RISC development going on at Apple. Two former members of defunct Ridge Computers (which used RISC technology)—one a founder, the other an R&D executive—have joined the Apple Corps.

Lord of the Rings
Supercomputer vendor Scientific Computer Systems is now offering the fastest (1.4 gigabytes per second) token ring local area network. Vectornet, as its called, also has the ability to gateway to other LANs.

Japan Inc.
Toshiba and NEC have both announced their working prototypes of charge-coupled device (CCD) image sensors in the 1035x1920 pixel resolution. The chip is 16.5 by 10 millimeters and will be available in commercial products within two years.

Sharp recently announced the availability of a 14-inch color LCD unit.

The recent Matsushita/Sun deal looks to be the start of some significant dealings with the Japanese. Apparently, Matsushita will build parallel-processing Sun-compatible workstations. According to my Japanese contact, the SPARC microprocessor is "very" attractive to the ever-growing Japanese computing industry. (The Tokyo University–created TRON architecture is starting to lose favor in Japan). I expect Fujitsu will soon follow suit with a low-end SPARC-based workstation. Don't expect it to stop there: Bill Joy has been spending a *lot* of time in Japan. I can think of several other companies in Japan that could make very profitable use of Sun's SPARC.

Don't forget that SPARC is *not* a chip; it is an architecture. Sun does not make a SPARC chip. Other companies license the architecture and fabricate hardware from the specifications.

A book publisher from Japan told me recently that the Japanese have an insatiable appetite for American software books. The most popular books in recent weeks have been K&R's C classic and texts on PostScript.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Does Sun Really Need AT&T?
A very prosperous workstation company like Sun does not need AT&T for the money. All the trade press are saying that Sun is going to use the "much-needed" cash infusion for R&D. What much-needed capital? Sun does not need AT&T's money. Just like the real reason that Sun wanted to consolidate their variety of Berkeley-tinged Unix (a really good one, I might add) and AT&T System V is for lucrative government bids. We're talkin' Big Bucks. Perhaps Sun wants access to AT&T's vast telecommunication customer base (PBXs, Large Central Office switches, wide area networking, and so on). Sun knows that the telecomm market is growing like hotcakes. With this information, Sun can now have the muscle to go against IBM and DEC. There is only one caveat: AT&T must leave Sun alone.

It really looks like Sun is taking over Wall Street. Many MS-DOS 386 machines are being replaced with Unix-powered Sun workstations; the Sun-4 seems especially attractive to many in the financial community. And it seems that the new 386i machine is also selling like hotcakes. Many of my clients have asked about its capabilities. Running DOS windows alongside Unix windows really comes in handy. It's really cute when you can open a named pipe on the 386i's Unix side, have a DOS process write to the pipe, and then have a Unix process read from the other end of the pipe.

Once the Windows driver is written for the Sun 386i, it could be a *very* attractive Windows development machine. After all, you need a serial terminal connected to your machine to debug MS Windows. With the 386i, you already have virtual terminals in SunView windows. So the serial "terminal" could be a second DOS window right next to your MS Windows window!

Build Memory Power
Many people misread Sun's recently-announced strategic agreement with Texas Instruments. All the trade rags pointed out how TI was going to use the SPARC processor in all their workstations. Who really cares about TI's workstations? Sun scored another coup by signing a *major* U.S. DRAM manufacturer on its team. Sun will obviously get preferential treatment when it comes to that scarce commodity called DRAMs. Not to mention TI's SPARC-compatible floating point chip and low-cost 34020 graphics processor.

A strong rumor has Sun merging their PC-NFS and TOPS networking software products. Many Macintosh owners would like the network transparency of NFS, but NFS does not (officially) exist for the Mac. TOPS is the only SUN network product available, although the two products overlap somewhat in the marketplace. Even though TOPS runs on AppleTalk and NFS runs over TCP/IP, it seems natural that these very successful products will eventually merge.

A Fujitsu representative told me recently that they have 40-MIPS versions of the SPARC architecture on the shelf ready to go (and Sun has samples in their labs). The reason for the delay (according to this techie): The agreement with Sun constricts Fujitsu from announcing new SPARCs "too early." Apparently, Sun feels that sales of their first-generation Sun-4 line (based on the Fujitsu MB86900 SPARC) might be affected negatively if potential customers hear of an even faster SPARC.

There are rumors that Perceptics, the VMS imaging vendor, is working on a machine called the Parallax, which uses several Sun's Sparc RISC processors and reaches supercomputer performance.

Judging from what I have seen and heard, I think the SPARC architecture will be a *big* winner.

NeWS News
Dave LaVallee, one of Sun's gifted NeWS programmers, appears to be headed for Sun's Advanced Technology Group. Based on his comments during the last few NeWS seminars, it looks as if he's implementing a hypermedia system based on NeWS. It seems that another NeWS aficionado and premier NeWS programmer, Don Hopkins of the University of Maryland, is porting HyperTIES, a hypermedia system, over to the Postscript-based window system.

Pop-up in the "In" Field
The trend in user interfaces suggests that the next generation will incorporate animation as a significant constituent. Notice that the "better" pop-up menus and windows don't just instantly appear on the screen. If user's blink their eyes, they may miss a window event. Newer computer animation for your local TV weather person is a widespread example—slowing down the action via clever animation so that the viewer sees the temperatures slowly appearing (sliding, sparkling, fading in) on the screen. While the techie wants his pop-ups to *snap* onto the screen in a microsecond, the end user is much more pleased with some ingenious animation. And besides, the animation (if it's done correctly) forces the user to concentrate.

Imperfect WordPerfect
According to a few in the know, there are two versions of the oft-touted WordPerfect 5.0. The version with the 5/88 date has a few bugs; the 7/88 version is the one you want. If you have the 5/88 version, the "maintenance version" is relatively cheap and may even be free, depending on when you purchased 5.0.

What's NeXT?
Clues to next year's software for Steve Job's NeXT machine are found in the past work done by the programming staff at Lucasfilms. Most of the programmers from Lucasfilms' "Droid Works" division now work for Mr. Jobs. Some of their very interesting work includes combining animated images, sound effects, and music on Sun Microsystems workstations. I saw an amazing demonstration of a visually oriented sound editor at the Portland Usenix created by Mike Hawley, et al.

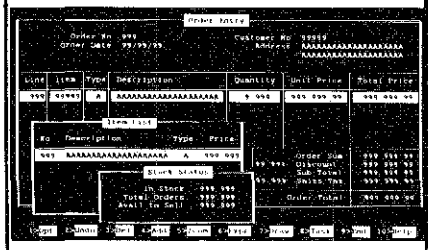
Rumors in the university circuit have it that Mr. Jobs is keenly interested in graphics pioneer Andries Van Dam's work with BALSA at Brown University.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Goodbye 4GL, Hello MAGIC PC

The *Visual* Database Language

TECH JOURNAL *Powerful DBMS applications in a fraction of the time."*



- **Powerful Application Generator**
- **BTRIEVE®-based Multiuser RDBMS**
- **Maintenance-Free and Code-Free**
- **Built-in Windows and Visual Query**

● 4GL's Are Too Slow

It's true. 4GL's are really "Slow GL's." If you program with dBASE®, Oracle®, etc., you know it takes forever to code, compile, debug and re-code a serious DBMS application.

● Develop 10 Times Faster

VAR'S, Consultants, DP-MIS and Power Users: try Magic PC to develop profitable turn-key systems like Accounting, Order Entry, Payroll, ***in days not months!***

Magic PC's Visual Language lets you concentrate ***only*** on your application design, and eliminate months of tedious coding. Draw your screens, windows, menus, reports, and define your complete programming logic with a minimal effort using the state-of-the-art Execution Tables. Magic PC's Runtime engine executes your design directly into finished programs without any coding. Even design changes are implemented automatically! Imagine finishing your application ***before*** conventional DBMS or 4GL programming begins. And you'll never waste another minute on coding or maintenance . . .

● Essential To BTRIEVE® Users

Magic PC is your instant application generator beyond Xtrieve® or Rtrieve® to create custom reports, screens, windows and complete front-ends, using your ***existing*** BTRIEVE® files!

Exclusive: Rent Before You Buy

Rent the ***full copy*** for 30 days for only $19.95. The rental fee applies to the special price offer:* Magic PC $199, Magic LAN $399 Multiuser, Magic Run — Call for price.

Dealers welcome.

Call now, rental copies are limited:

800-345-MAGIC

in CA 714-250-1718

AKER

19782 MacArthur Bl. Suite 305, Irvine, CA 92715

FAX: 714-955-0199 Telex: 493-1184

DS CL DB DD PT IN PU XX PS

* Add $10 P&H, tax in CA. International orders add $30. Rental offer valid in US only, subject to availability. Requires IBM/100% comp., 512K, hard disk, DOS 3.0 or later. Includes BTRIEVE runtime. 5.25" disk, not copy protected. Copyright © Aker, 1988.

Number **116** on the Reader Service Card

It also seems that Jobs wants to do Apple's Hypercard one better by incorporating hypermedia elements into the NeXT workstation's user interface.

Similar to Sun's strategy, MIPS Computer Systems has decided to license the manufacturing of their processors. LSI Logic, Integrated Device Technology, Inc., and Performance Semiconductor Corp. are now licensed to manufacture the MIPS R2000 RISC-based processor. Performance Semiconductor has promised a 20-MIPS version of the R2000 by the end of 1988.

Da Vinci Systems Corporation of Raleigh, North Carolina, (near Research Triangle) has extended Microsoft's Dynamic Data Exchange (DDE) across networks. Their software product, called "Dispatcher," uses the NetBIOS protocol and will—it's still in development—cost less than $100. This will allow processes to send data dynamically to any other active process on a network. While new to the MS-DOS world, such interprocess communication has long been done in the Unix world using "sockets".

Expect to see very-high-density floppy disk drives available by early 1989. Brier Technology (408-435-8463) promises a 20-megabyte, 35-ms, 3.5-inch drive that will use new high density Barrite media 3.5-inch "floppy" disks. Irwin Magnetic Systems will OEM the technology in a commercial product during the first quarter. Brier's 40-megabyte version will be available sometime in mid 1989.

I noticed that the TI 34010 does not do arbitrary polygon clipping, only rectangular; so it may not be that applicable for a PostScript Raster Image Processor (RIP).

From Math Maestros to PostScript

I saw Weitek's newest Postscript hardware recently and was thoroughly impressed. This board renders Postscript images so fast that it must be slowed when communicating with the laser engine. I saw the board connected to an HP LaserJet—amazing! The board processed complex graphics and multifont text at the rated speed of the laser engine. And this is a first-generation product! Rumors have it that Weitek's hardware will be used by Adobe in the newest "Hercules" PostScript controller. The Weitek people have mentioned that they are developing a version that assists Display PostScript in rendering PostScript images on screen. I would bet money that these people are also talking to Sun regarding NeWS, since it too is PostScript based.

Very soon someone will be announcing a tabletop typesetter with resolution of up to 5040 dpi for just over $10,000. While the ordinary letter writer does not need this resolution, small print shops could compete (somewhat) with the big shops.

Lotus Pay Our Taxes

According to the *WSJ* dated 3/23/87, Jim Manzi, the president of Lotus Development Corporation, stated that he had to sell 250,000 shares of Lotus stock to cover his taxes. At the current rate of $28.75 per share, that's $7,187,500 just to cover his taxes!

Wholly Holography!

Hughes Research Lab has managed to store images in holographic format. Using optical associative memory, neural network software, and holography, these scientists are able to reconstruct a complete image from only 20% of the image's data.

AT&T has announced that Sun's Remote Procedure Call (RPC) and Network File System (NFS) will be included in System V Version 4. Also, some features from Berkeley Unix will be included (I pray that job control will be one). I'm waiting for the day that all Unixs have a standard RPC mechanism; this announcement just makes it a matter of time.

Thanks for the Memories

At the last International Solid State Circuits Conference, Toshiba, Hitachi, and Matsushita all discussed 16-megabit DRAMs. The companies seemed confident that production quantities would be available by 1990. Let's hope the Japanese/American DRAM situation will be resolved by then.

At the same conference, General Electric and Matsushita both described graphics processors that do hidden-line removal for 3-D applications. Toshiba described their new BITBLT chip, which does 40 megapixels per second. Just to give you an idea of the speed of this chip, it can redraw an entire 256-color 1024x1024 screen in .025 seconds.

A Waste Is a Terrible Thing to Mind

AT&T's Jack Scanlon, when asked why the price and performance of a RISC processor is better than that of a CISC processor: "An Intel 80386 chip today costs $300—even if I put all of New York

PRODUCED 2004 BY GNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

City on the chip." OK, Jack (the doctor says to play along with him).

That's Logical
One of Sun's biggest OEM's, Toshiba, is going to put their IP704 AI microprocessor into every Sun workstation that they resell. Supposedly, this processor can run LISP, Prolog, or C and can perform at 600 KLIPS (Logic Inferences Per Second). Toshiba calls this Sun-based workstation the AS4000.

Call Ripley
Isn't it a really weird coincidence that when you reverse the name of the inventor of DEC's DECTALK voice synthesizer, MIT's Klatt, who has a speaking disability himself, you get ttalk! Weird.

Getting Wired
Edge Computers, of Scottsdale, Arizona, has implemented the architecture of the Motorola 680X0 using high-speed gate arrays, hardwired control logic, advanced pipelining, and nifty caching techniques. By doing this, Edge has greatly enhanced the performance of the microprocessor up to 16 MIPS. If you connect four of these processors together, you can get a sustained 55 MIPS. Edge is not a second source of the Motorola part; they are only producing a powerful computer engine that is instruction-set compatible with the 680x0. Apparently, the "Edge-ites" are also currently working on 80286 and 80386 versions.

Twisted Pair
DEC and Radio Shack????? I'm sorry, but I don't see the advantage that DEC derives from this marriage.

Did-You-Know Dept.
Interactive Systems, which is porting AIX for IBM (and DEC and HP and Apollo), is owned by Kodak.

Soda Pop Will Fizzle Out
Is the real reason that Apple's John Sculley is suing Microsoft and HP because he wants to license the Apple Macintosh interface to the PC world? Rumors abound of Mr. Sculley's indication to the Apple spin-off, Claris, that a Macintosh programmers' toolkit written and licensed for the PC world would be a big moneymaker for Apple. Whatever the real reason, I think that Apple's lawsuit will eventually come back to haunt the corporation.

continued . . .

PERSONAL REXX 2.0
NEW PERSONAL PROGRAMMING POWER

Personal REXX is a powerful general-purpose language bringing mainframe command and macro language capabilities to the PC. Personal REXX 2.0 is a new, faster version that runs larger programs, and features:

- **A superset of IBM CMS REXX**
- **The ability to develop powerful batch procedures**
- **Compatibility with IBM SAA REXX command language**
- **Windowing and menu support**
- **Easy prototyping**
- **The ability to interface with KEDIT™ text editor and HyperTerm™ communication package**

Personal REXX runs on the IBM PC, AT, PS/2, and true compatibles using DOS 2.0 or later. A Programmer's Journal review said: ***"Everyone who tries it seems to love it."*** Call to find out why, and how you can enjoy the programming power and the many benefits of Personal REXX 2.0

Personal REXX Version 2.0 is available at $150. Add $3 shipping. MC, VISA, American Express.

MANSFIELD
Software Group
P.O. Box 532, Storrs, CT 06268
(203) 429-8402

KEDIT is a trademark of the Mansfield Software Group, Inc.; HyperTerm is a trademark of Quercus Systems; IBM and IBM PC, AT and PS/2 are trademarks of International Business Machines.

Number **233** on the Reader Service Card

TRUE MULTITASKING
With
MultiDos Plus
"multitasking for the IBM-PC."

Ideal for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 programs concurrently.
- Your software continues to run under DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in most languages.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/resume programs.
- Programmatic interface via INT 15H for the following.
  - * Intertask message communication. Send/receive/check message present on 64 message queues.
  - * Task control by means of semaphores. Get/release/check semaphores.
  - * Change priority-256 priority levels.
  - * Suspend task for specified interval.
  - * Spawn and terminate external and internal tasks.
  - * Disable/enable multitasking.
  - * and more!
- Independent foreground/background displays.
- Access to DOS while applications are running.

Hardware/Software Requirements

IBM PC/XT/AT or true clone. Enough memory to hold **MultiDos Plus** (48 KB) and all your application programs. Also may need 4 or 16 KB memory for "hidden screens" for each active task. MS-DOS (or PC-DOS) 2.0 or later operating system.

only: $24.95 *OR* $99.95 with source code

Outside USA add $5.00 shipping and handling.
Visa and Mastercard orders only call toll-free: 1-800-872-4566, ext. 350., or send check or money order to:

NANOSOFT
13 Westfield Rd, Natick, MA 01760
MA orders add 5% sales tax.

Number **109** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Optical Info

Verbatim, owned by Eastman Kodak, will soon announce an erasable 3.5-inch optical disk drive. Information that I received from a recent optical technology conference shows Verbatim about a year away from releasing this commercially. The drive will use a half-height format and thermo-magneto optical technology. Representatives told me to expect the first-generation product to have 50- to 100-megabyte capacity, 70-ms access times, and 1-megabyte-per-second transfer rates.

Apparently these optical devices will be used as floppy disk replacements, although their ability to replace winchester hard drives is up in the air. Hybrid vertical recording techniques appear to stretch the life of winchesters a little longer than previously expected. Representatives from such companies as IBM and new start-up Censtor are predicting 100-megabyte hard cards within a year and 1-gigabyte external drives within two years.

NEC's new PD72185 image compression/reconstruction chip for black-and-white images can perform compression or uncompression on a standard CCITT test page in less than .75 seconds. Expect to see this chip in all new facsimile boards; it can handle the three most popular compression and reconstruction algorithms for faxes. This chip could also be very useful in image workstation environments. An image that was stored on disk in CCITT group-3 or group-4 format could be recalled from disk, expanded, and displayed in less than a second.

Usenix

At a recent Usenix, I talked to a old, former-Bell-Labbie friend of mine. He told me that the prime developer of the C on a RISC Processor (CRISP), who was frustrated with the bureaucracy of AT&T in marketing his invention, is now employed by Sun Microsystems. This researcher is currently responsible for a Gallium Arsenide version of the Sun SPARC processor.

One of C++'s prime proponents, Andrew Koenig of AT&T Bell Laboratories, appears to be readying a book on "C traps and pitfalls," based on the subject's recent community brain dump over Usenet.

I also heard at Usenix that one of Sun's founders, hardware guru Andy Bechtolsheim, is hard at work on a machine that is targeted for the masses. Apparently, he has been working on this machine for the past two years (rumors started at the 1987 Usenix Conference in Phoenix). Codenamed "campus," this machine supposedly lies in the $3,000 to $5,000 price bracket. Is this the SPARCintosh?

Remember These Words

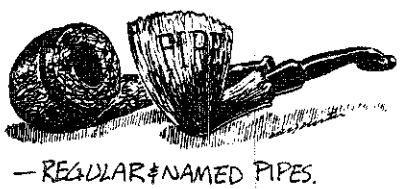
Ken Bosomworth of IRD: "I think Unix will fade away, but it may take a decade."

Amy Wohl: "At the user level I do not think that Unix will be a standard. It would be hard to list 10 or 15 good Unix workstation packages outside the field of engineering."

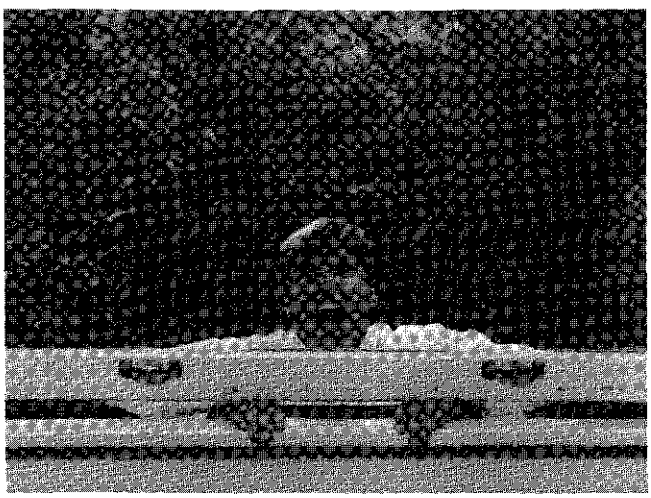
These people are dead wrong. ♦

Frank D. Greco is an independent consultant who offers industry analysis, systems consulting, and technological forecasting for Fortune 500 companies. Readers can contact Frank at the following address and phone number:

Frank D. Greco, c/o Crossroads Technologies, Inc., P.O. 530, Fanwood, New Jersey 07023 (201) 754-7820



Feeling trapped by your windowing software?



Frustrated with the limitations of your windowing package?

ÆWINDOS is a high-powered set of tools designed to interactively create windows at light speed with both DOS and UNIX systems.

Written entirely in portable "C", it includes the complete source code and even make files, so changes are quick and easy.

Don't wait until you're caught in a bind, call us today. Modular, portable, and efficient, ÆWINDOS leaves you open to possibilities.

æSOFT

1-800-634-5494 BBS (303) 449-5714

Number **220** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C CODE FOR THE PC

source code, of course

	MS-DOS File Compatibility Package (create, read, & write MS-DOS file systems on non-MS-DOS computers)	$500
	Bluestreak Plus Communications (two ports, programmer's interface, terminal emulation)	$400
	CQL Query System (SQL retrievals plus windows)	$325
	GraphiC 4.1 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	$325
	PC Curses (Aspen, Software, System V compatible, extensive documentation)	$250
NEW!	Greenleaf Business Mathlib (exact decimal math, formatting, depreciation, interest, cash flow, statistics)	$250
	Greenleaf Data Windows (windows, menus, data entry, interactive form design)	$220
NEW!	Assembler Kit (by John Zarrella; includes listing generator & loader; requires signed license agreement)	$175
	Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF)	$175
	TurboTEX (TRIP certified; HP, PS, dot drivers; CM fonts; LaTEX)	$170
NEW!	Sherlock (C debugging aid)	$170
	Greenleaf Functions (296 useful C functions, all DOS services)	$160
	Essential C Utility Library, Communications Library, or Resident C	$160
	Essential Communications Library (C functions for RS-232-based communication systems)	$160
	WKS Library Version 2.0 (C program interface to Lotus 1-2-3, dBase, Supercalc 4, Quatro, & Clipper)	$155
	OS/88 (U**x-like operating system, many tools, cross-development from MS-DOS)	$150
	ME Version 2.1 (programmer's editor with C-like macro language by Magma Software; Version 1.31 still $75)	$140
	Vmem/C (virtual memory manager; least-recently used pager; dynamic expansion of swap file)	$140
	Turbo G Graphics Library (all popular adapters, hidden line removal)	$135
	TurboGeometry (library of routines for computational geometry)	$125
	CBTree (B+tree ISAM driver, multiple variable-length keys)	$115
	Minix Operating System (U**x-like operating system, includes manual)	$105
	PC/IP (CMU/MIT TCP/IP implementation for PCs)	$100
	B-Tree Library & ISAM Driver (file system utilities by Softfocus)	$100
	Tele Operating System (TeleKernel, TeleWindows, TeleFile, & TeleBTree by Ken Berry)	$100
	The Profiler (program execution profile tool)	$100
	QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	$90
	Wendin Operating System Construction Kit or PCNX, PCVMS O/S Shells	$80
	C Windows Toolkit (pop-up, pull-down, spreadsheet, CGA/EGA/Hercules)	$80
	JATE Async Terminal Emulator (includes file transfer and menu subsystem)	$80
	Polyglot Lisp-to-C Translator (includes Lisp interpreter, Prolog, and simple calculus prover)	$80
	MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores)	$80
NEW!	Kinetic Image Synthesizer (3-D animation system ... Saturday morning on your PC!)	$75
	XT BIOS Kit (roll your own BIOS with this complete set of basic input/output functions for XTs)	$75
	TE Editor Developer's Kit (full screen editor, undo command, multiple windows)	$75
	Professional C Windows (lean & mean window and keyboard handler)	$70
NEW!	Heap Expander (use LIM-standard expanded memory as an extension of the heap)	$65
	lp (flexible printer driver; most popular printers supported)	$65
	Quincy (interactive C interpreter)	$60
	PTree (parse tree management)	$60
NEW!	OBJASM (convert .obj files to .asm files; output is MASM compatible)	$50
NEW!	Icon-Tools (full-featured icon display and editing system)	$60
NEW!	Polyglot TSR Package (includes reminder, bookmark, virus catcher, cache manager, & speech generator)	$50
	HELP! (pop-up help system builder)	$50
	Multi-User BBS (chat, mail, menus, sysop displays; uses Galacticomm modem card)	$50
	Make (macros, all languages, built-in rules)	$50
	Coder's Prolog (inference engine for use with C programs)	$45
	Virtual Memory System (least recently used swapping)	$40
	C-Notes (pop-up help for C programmers ... add your own notes)	$40
	Heap I/O (treat all or part of a disk file as heap storage)	$40
	Biggerstaff's System Tools (multi-tasking window manager kit)	$40
	PC-XINU (Comer's XINU operating system for PC)	$35
	CLIPS (rule-based expert system generator, Version 4.2)	$35
	Tiny Curses (Berkeley curses package)	$35
NEW!	Polyglot RAM Disk (change disk size on the fly; includes utilities)	$30
	SP (spelling checker with dictionary and maintenance tools)	$30
	Clisp (Lisp interpreter with extensive internals documentation)	$30
	Translate Rules to C (YACC-like function generator for rule-based systems)	$30
	6-Pack of Editors (six public domain editors for use, study & hacking)	$30
	Crunch Pack (14 file compression & expansion programs)	$30
NEW!	Install (automatic installation program; user-selected partial installation; CRC checking)	$25
	Pascal Compiler & Interpreter or Pascal-to-C Translator (P-codes, standard Pascal)	$25
	ICON (string and list processing language, Version 7.5)	$25
	FLEX (fast lexical analyzer generator; new, improved LEX)	$25
	LEX (lexical analyzer generator; an oldie but a goodie)	$25
	Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor)	$25
	AutoTrace (program tracer and memory trasher catcher)	$25
	Data Handling Utilities in C (data entry, validation & display; specify Turbo C or Microsoft)	$25
	Arrays for C (macro package to ease handling of arrays)	$25
	ANSI Forms (forms manager based on ANSI codes)	$20
	C Compiler Torture Test (checks a C compiler against K & R)	$20
	Benchmark Package (C compiler, PC hardware, and Unix system)	$20
	A68 (68000 cross-assembler)	$20
	List-Pac (C functions for lists, stacks, and queues)	$20
	XLT Macro Processor (general purpose text translator)	$20
	C/reativity (Eliza-based notetaker)	$15
	Data	
	DNA Sequences (GenBank 55.0 including fast similarity search program)	$150
	Protein Sequences (5,415 sequences, 1,302,966 residuals, with similarity search program)	$70
NEW!	SIC Codes (each SIC code with a description of the industry to which it applies)	$70
	Dictionary Words (234,932 words in alphabetical order, no definitions)	$60
	U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	$35
	The World Digitized (100,000 longitude/latitude of world country boundaries)	$30
	KST Fonts (13,200 characters in 139 mixed fonts: specify TEX or bitmap format)	$30
	USNO Floppy Almanac (high-precision moon, sun, planet & star positions)	$20
	NBS Hershey Fonts (1,377 stroke characters in 14 fonts)	$15
	U. S. Map (15,701 points of state boundaries)	$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409 USA

acw!info@uunet.uu.net

Voice: (512) 258-0785
BBS: (512) 258-8831
FAX: (512) 258-1342

Free surface shipping on prepaid orders **For delivery in Texas add 7%** **MasterCard/VISA**

Number **30** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Higher 256-Color Resolution on the VGA

by
Michael Abrash

Installment 10: In which we find the answer to the riddle "When is 320x200 really 320x400?" and chance upon some intriguing 256-color resolutions.

One of the more appealing features of the VGA is the ability to display 256 simultaneous colors. Unfortunately, one of the less appealing features of the VGA is the limited resolution (320x200) of the one 256-color mode the BIOS supports. More color can often compensate for less resolution, but the resolution difference between the 640x480 16-color mode and the 320x200 256-color mode is so great that many programmers must regretfully decide that they simply can't afford to use the 256-color mode.

If there's one thing we've learned about the VGA, however, it's that there's *never* just one way to do things. With the VGA, alternatives always exist for the clever programmer, and that's more true than you might imagine with 256-color mode. Not only is there a high 256-color resolution, there are *lots* of higher 256-color resolutions, going all the way up to 360x480! (Here I'm talking about the vanilla IBM VGA—third-party VGAs go all the way up to 800x600 256-color resolutions, but that's a story for another time.)

While I'd love to cover all of the 256-color resolutions, high-resolution 256-color programming is a fairly complex topic, and I've only so much space and time. Consequently, I'm going to take a quick look at the available 256-color resolutions and then focus on my personal favorite 256-color mode, which provides 320x400 resolution and two graphics pages and can be set up with very little reprogramming of the VGA. So. Let's get started.

Why 320x200? Only IBM Knows for Sure

The first question is, of course, How can it be possible to get higher 256-color resolutions out of the VGA? After all, there were no unused higher resolutions to be found in the CGA, Hercules card, or EGA.

The answer is another question: Why did IBM *not* use the higher-resolution 256-color modes of the VGA? The VGA is easily capable of twice the 200-scan-line vertical resolution of mode 13h, the 256-color mode, and IBM clearly made a decision not to support a higher-resolution 256-color mode. In fact, mode 13h *does* display 400 scan lines, but each row of pixels is displayed on two successive scan lines, resulting in an effective resolution of 320x200. This is the same scan-doubling approach used by the VGA to convert the CGA's 200-scan-line modes to 400 scan lines; however, because the resolution of the CGA has long been fixed at 200 scan lines, IBM had no choice with the CGA modes but to scan-double the lines. Mode 13h has no such historical limitation—it's the first 256-color mode ever offered by IBM, if you don't count the late and unlamented PGC. Why, then, would IBM choose to limit the resolution of mode 13h?

There's no way to know, but one good guess is that IBM wanted a standard 256-color mode across all PS/2 computers, and mode 13h is the highest-resolution 256-color mode that could fill the bill. You see, each 256-color pixel requires one byte of display memory, so a 320x200 256-color mode requires 64,000 bytes of display memory. That's no problem for the VGA, which has 256K of display memory, but it's a stretch for the MCGA of the Model 30, since the MCGA comes with only 64K.

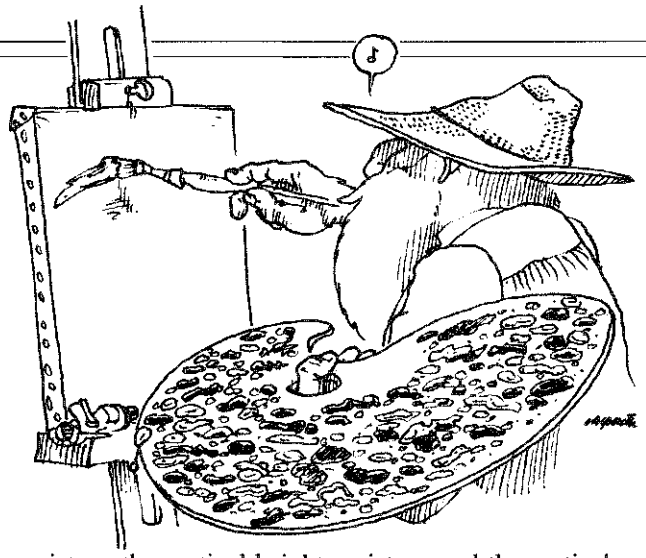
On the other hand, the smaller display-memory size of the MCGA also limits the number of colors supported in 640x480 mode to 2, rather than the 16 supported by the VGA. In this case, though, IBM simply created two modes and made both available on the VGA: mode 11h for 640x480 2-color graphics and mode 12h for 640x480 16-color graphics. The same could have been done for 256-color graphics—but it wasn't. Why? I don't know. Maybe IBM just didn't like the odd aspect ratio of a 320x400 graphics mode. Heck, maybe they made a mistake in designing the chip. Whatever the reason, mode 13h is really a 400-scan-line mode masquerading as a 200-scan-line mode, and we can readily end that masquerade.

Higher 256-Color Resolutions

The VGA can actually display even higher 256-color resolutions than 320x400. Resolutions of 320x480, 360x400, and 360x480 are all possible. I'm going to take a quick look at each of the three higher resolutions and then return to the 320x400, which I think is the best of the lot.

The number of scan lines on the screen is a function of the number of vertical frames displayed each second, which in turn is controlled by the vertical timing registers of the CRT Controller (CRTC). Most VGA modes, including mode 13h, display 70 frames per second, with each frame 400 scan lines high. However, modes 11h and 12h both display sixty 480-scan-line frames per second. Getting 480 scan lines out of mode 13h (320x480 256-color mode) is a simple matter of selecting the mode and then reprogramming the vertical timing

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



registers, the vertical height registers, and the vertical and horizontal sync polarities along the lines of mode 12h. (The sync polarities tell the VGA monitor how rapidly to scan vertically. Slower scanning is needed to allow 480 scan lines to fit on the screen.)

360x400 256-color mode requires a change in the horizontal scanning rate of the VGA. You accomplish this by selecting the faster VGA clock in the Miscellaneous Output register. This clock is normally used for scanning 720 pixels across the screen in 9-dots-per-character 80-column text modes but will serve equally well to scan 720 pixels across the screen in graphics mode in the form of ninety 8-dot characters. Of course, changing the clock speed and the number of characters per line requires reprogramming of the horizontal timing and horizontal width registers in the CRTC to match, just as changing the frame rate and number of scan lines per frame required reprogramming of the vertical timing registers. Since there is no 90-column 8-dot-per-character mode, these parameters must be calculated from scratch or by trial and error.

With VGA monitors, horizontal timings and vertical timings operate independently; so, all you need to do to get 360x480 resolution is combine the 320x480 and 360x400 256-color modes. All of these modes also require reprogramming in order to let your programs get at display memory properly, as we'll see shortly in our discussion of the 320x400 mode.

You've surely noticed that I haven't given any specifics about register values and such for selecting these higher-resolution modes. There's a good reason for that: Each of these modes comes with certain risks for the developer. For one thing, the modes with 360 pixels per scan line run the VGA's memory faster than any other VGA mode, and I don't know if that access speed is within the specs of the VGA's memory. For another, there's always danger in reprogramming the timing registers. Sure, every VGA clone claims register compatibility, but it's on just such nonstandard modes as these that compatibility tends to break.

Given the risks of the above modes, given that I have limited space, *and* given that there's a mode I consider safe and superior, I'm not going to pursue the three higher-resolution 256-color modes I've mentioned any further, at least not in this installment. Those of you who really want to use these modes should be able to figure out what to do from the thumbnail sketches I've given. If enough readers request it, however, I would be happy to devote a future installment to these three modes.

By the way, I'm probably overstating the risks of the three higher-resolution 256-color modes. I read recently that Rix is coming out with a 360x480 256-color program for the VGA. Those guys know the VGA awfully well, and they have access to just about every VGA clone in existence. If they think 360x480 256-color mode is safe enough to use in a commercial product, it's a pretty safe bet that it is.

Still and all, I like 320x400 256-color mode best. 320x400 requires very little VGA reprogramming (and none at all to the timing registers) and gives us two pages—ideal for animation and flicker-free drawing. So, it's onward to 320x400 and working code!

320x400 256-Color Mode

Okay, what's so great about 320x400 256-color mode? Two things: easy, safe mode sets and paging.

As I said above, mode 13h is really a 320x400 mode, albeit with each line doubled to produce an effective resolution of 320x200. That means that we don't need to change any display timings, widths, or heights in order to tweak mode 13h into 320x400 mode—and that makes 320x400 a safe choice. Basically, 320x400 mode differs from mode 13h only in the settings of *mode* bits, which are sure to be consistent from one VGA clone to the next and which work equally well with all monitors. The other hi-resolution 256-color modes differ from mode 13h not only in the settings of the mode bits but also in the settings of timing and dimension registers, which may not be exactly the same on all VGA clones and particularly not on all multisync monitors. (Because multisyncs can sometimes shrink or expand VGA displays, some VGAs may use alternate register settings for multisync monitors that are optimized for those monitors.)

The other good thing about 320x400 256-color mode is that two pages are supported. Since each 320x400 256-color mode requires 128,000 bytes of display memory, we can just barely manage two pages in 320x400 mode, one starting at offset 0 in display memory and the other starting at offset 8000h. Those two pages are the largest pair of pages that can fit in the VGA's 256K, though, and the higher-resolution 256-color modes, which use still larger bitmaps, can't support two pages at all. As we've seen in earlier installments of "On Graphics," paging is very useful for off-screen construction of images and fast, smooth animation.

That's why I like 320x400 256-color mode. The next step is to understand how the bitmap is organized in 320x400 mode, and that's not so simple.

Display-Memory Organization in 320x400 Mode

First, let's look at why display memory must be organized differently in 320x400 256-color mode than in mode 13h. The designers of the VGA intentionally limited the maximum size of the bitmap in mode 13h to 64K, thereby limiting resolution to 320x200. This was accomplished *in hardware*, so there is no way to extend the bitmap organization of mode 13h to 320x400 mode.

That's a shame, because mode 13h has the simplest bitmap organization of any mode—one long, linear bitmap, with each byte controlling one pixel. We can't have that organization,

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

though; so we'll have to find an acceptable substitute if we want to use a higher 256-color resolution.

With the VGA, of course, there are actually *several* bitmap organizations that let us use higher 256-color resolutions than mode 13h. The one I like best is shown in Figure 1. Each byte controls one 256-color pixel. Pixel 0 is at address 0 in plane 0, pixel 1 is at address 0 in plane 1, pixel 2 is at address 0 in plane 2, pixel 3 is at address 0 in plane 3, pixel 4 is at address 1 in plane 0, and so on.

Let's look at this another way. Ideally, we'd like one long bitmap, with each pixel at the address that's just after the address of the pixel to the left. Well, that's true in this case too, *if* you consider the number of the plane that the pixel is in to be part of the pixel's address. View the pixel numbers on the screen as increasing from left to right and from the end of one scan line to the start of the next. Then the pixel number, *n*, of the pixel at display-memory address *address* in plane *plane* is

$$n = (address * 4) + plane$$

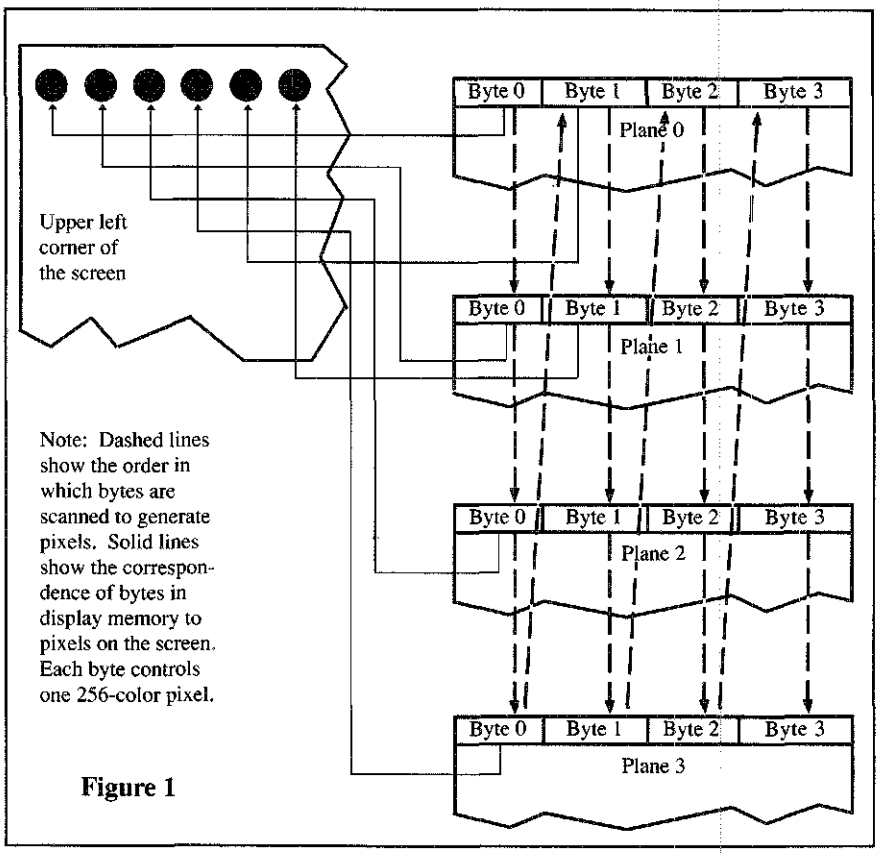

Byte 0
Byte 1
Byte 2
Byte 3
Plane 0
Plane 1
Plane 2
Plane 3
Upper left corner of the screen
Note: Dashed lines show the order in which bytes are scanned to generate pixels. Solid lines show the correspondence of bytes in display memory to pixels on the screen. Each byte controls one 256-color pixel.

Figure 1

To turn that around, the display-memory address of pixel number *n* is given by

$$address = n / 4$$

and the plane of pixel *n* is given by

$$plane = n \text{ modulo } 4$$

Basically, the full address of the pixel, its pixel number, is broken into two components: the display-memory address and the plane.

By the way, because 320x400 mode has a significantly different memory organization from mode 13h, the BIOS text routines won't work in 320x400 mode. If you want to draw text in 320x400 mode, you'll have to look up a font in the BIOS ROM and draw the text yourself. Likewise, the BIOS read pixel and write pixel routines won't work in 320x400 mode, but that's no problem: I'll provide equivalent routines in the next section.

Our next task is to convert standard mode 13h into 320x400 mode. That's accomplished by undoing many of the mode bits that are set up especially for mode 13h so that from a programming perspective the VGA reverts to a straightforward planar model of memory. That means taking the VGA out of chain-4 mode and doubleword mode, turning off the double display of each scan line, making sure chain mode, odd/even mode, and word mode are turned off, and selecting byte mode for video data display. All that's done in the Set320x400Mode subroutine in Listing 1, which we'll discuss next.

Reading and Writing Pixels

The basic graphics functions in any mode are functions to read and write single pixels. Any more-complex function can be built on these primitives, although that's not always the speediest solution. What's more, once you understand the operation of the read and write pixel functions, you've got all the knowledge you need to create functions that perform more-complex graphics functions. Consequently, we'll start our exploration of 320x400 mode with pixel-at-a-time line drawing.

Listing 1 draws eight multicolored octagons, drawing a new one on top of the old one each time a key is pressed. The main-loop code of Listing 1 should be easy to understand: A series of diagonal, horizontal, and vertical lines are drawn one pixel at a time based on a list of line descriptors, with the draw colors incremented for each successive time through the line list.

Three aspects of Listing 1 are particularly interesting:

1. The Set320x400Mode subroutine selects 320x400 256-color mode by performing a mode-13h mode set and then putting the VGA into standard planar byte mode. Set320x400Mode zeros display memory as well. It's necessary to clear display memory even after a mode-13h mode set, because the mode-13h mode set clears only the 64K of display memory that can be accessed in that mode, leaving 192K of display memory untouched.

2. The WritePixel subroutine draws a colored pixel at any *x,y* addressable location on the screen. Although it may not be obvious because I've optimized the code a little, the process of drawing a pixel is remarkably simple. First, the pixel's display-memory address is calculated as

$$address = [x * (SCREEN\_WIDTH / 4)] + (y / 4)$$

which might be more recognizable as

$$address = [(x * SCREEN\_WIDTH] + y) / 4$$

continued . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

INGRAF 2.00

Full Source code $295.00

The multi-device **GRAPHICS** library for scientific/ Engineering and Busines Graphics applications.**INGRAF** contains over **100 routines.**

Features include:

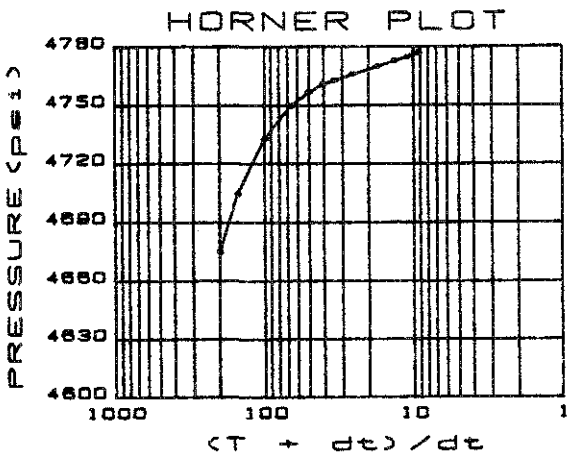
- *Linear, Semilog, Log/Log, Polar plots.*
- *Axes and Grids with tick marks and labels.*
- *Bar, Pie, Smith charts.*
- *Markers, Line types, Curves, Arcs, Circles, Ellipses.*
- *Filling and Hatching of bounded areas.*

Supports:

- *CGA, EGA, VGA, HERCULES Graphics cards.*
- *HPGL, DM/PL pen plotters.*
- *HP LaserJet, HP DeskJet, Epson, Okidata printers.*

Languages Supported:

- **C**
- **FORTRAN-77**
- **BASIC**
- **PASCAL**



Compilers Supported:

C	FORTRAN-77	BASIC	PASCAL
Microsoft C	MS FORTRAN	Quick Basic	MS Pascal
Turbo C	RM FORTRAN	Turbo Basic	Turbo Pascal
Lattice C	Lahey F77L		
MWC-86	IBM Prof. FORTRAN		

FORTLIB 4.00

Full Source code $150.00

The **FORTRAN-77** Enhancement library for **IBM-PC (MS-DOS/PC-DOS)** and compatibles. The library contains well over **200 routines.** Most routines are written in assembly for speed and compact code.

Features include:

- *Keyboard handling.*
- *Video setup and control.*
- *Date and time functions.*
- *Disk directory and file handling.*
- *General I/O port functions.*
- *Memory PEEK and POKE functions.*
- *Data sorting.*
- *Extensive string handling.*
- *Data conversion routines.*
- *Character classification functions.*
- *RS-232 service routines.*
- *Mepory manipulation,*
- *printer and system service routines.*
- *and many miscellaneous routines.*

GRAFLIB 4.00 Full Source code $175.00

Video Graphics Library for **IBM-PC** and compatibles. The library supports **CGA, EGA, VGA** and **HERCULES** Graphics cards. There are over 50 routines for drawing **Arcs, Circles, Line drawing, Axes, Labeling of text, Curves, Line types, Markers** to name a few.

The library is available in **C** & **FORTRAN-77** bindings.

PLOTHP 4.00 Full Source code $175.00

Plotting routines for **HPGL(Hewlett-Packard)** compatible plotters. The Library contains over **60 routines** for drawing **Arcs, Circle, Curves, Linear, Semilog, Log/Log** plots. **Axes drawing with tick marks, Text labeling, Scaling of plots** is simple with **PLOTHP** routines. The library includes routines for specifying **Line types, Marker types, Text direction** and **Orientaton.**

The Library is available in **C** & **FORTRAN-77** bindings.

For more information and a detailed brochure please call or write to:

Sutrasoft (713) 491-2088

10506 Permian Dr.
P.O. Box 1733
Sugar Land, TX 77487-1733

Number 253 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

(There are 4 pixels at each display-memory address in 320x400 mode, hence the division by 4.) Then, the pixel's plane is calculated as

$$plane = y \text{ and } 3$$

which is equivalent to

$$plane = y \text{ modulo } 4$$

The pixel's color is then written to the addressed byte in the addressed plane. That's all there is to it!

3. The ReadPixel subroutine is virtually identical to WritePixel, save that in ReadPixel the Read Map register is programmed with a plane number, while WritePixel uses a plane *mask* to set the Map Mask register. Of course, that merely reflects a fundamental difference in the operation of the two registers. (If that's Greek to you, you'll want to get back issues 5.1-5.4, 6.1, 6.2, and 6.4-6.6 of *Programmer's Journal* and come up to speed on the basics of EGA and VGA programming.) ReadPixel isn't used in Listing 1, but I've included it because, as I said above, the read and write pixel functions together can support a whole host of more-complex graphics functions.

How does 320x400 256-color mode stack up in regard to performance? As it turns out, the programming model of 320x400 mode is actually quite good for pixel drawing—very nearly on a par with the model of mode 13h. When you run Listing 1, you'll no doubt notice that the lines are drawn quite rapidly. (In fact, the drawing could be considerably faster still with a dedicated line-drawing subroutine, which would avoid the multiplication associated with each pixel in Listing 1.)

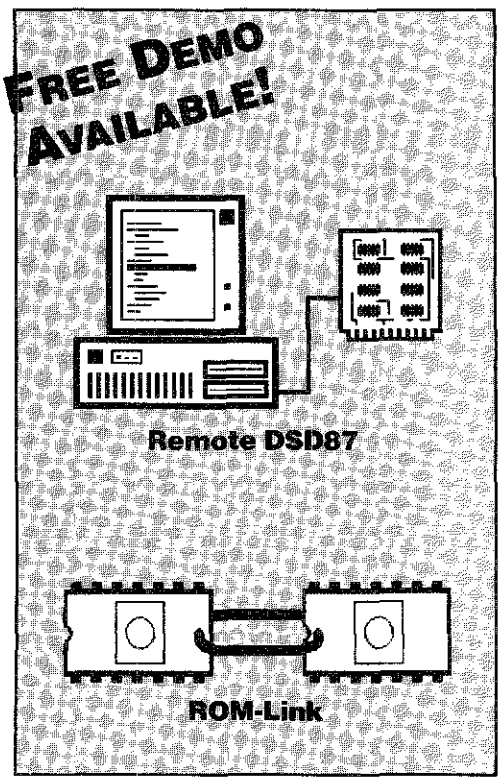
In 320x400 mode, the calculation of the memory address is not significantly slower than in mode 13h, and the calculation and selection of the target plane is quickly accomplished. As with mode 13h, 320x400 mode benefits tremendously from the byte-per-pixel organization of 256-color mode, which eliminates the need for the time-consuming pixel masking of the 16-color modes. Most important, byte-per-pixel modes never require read/modify/write operations (which can be extremely slow owing to display-memory wait states) to clip and draw pixels. To draw a pixel, you just store its color in display memory. What could be simpler?

More-sophisticated operations than pixel drawing are less easy to accomplish in 320x400 mode, but with a little ingenuity it is possible to implement an efficient version of just about any useful graphics function. A fast line draw for 320x400 mode would be simple, and fast image copies could be implemented by copying one quarter of the image to one plane, one quarter to the next plane, and so on for all four planes. If you're really into performance, you could store your images with all the bytes for plane 0 grouped together, followed by all the bytes for plane 1, and so on. That would allow a single rep movs instruction to copy all the bytes for a given plane, with just four rep movs instructions copying the whole image.

It's all a bit complicated, but as I say, you should be able to design a fast version for 320x400 mode of whatever graphics function you need. If you're not all that concerned with speed, WritePixel and ReadPixel should meet your needs.

continued . . .

For the Ultimate in Embedded Systems Programming.



Debug Debug Debug Debug Debug

Remote DSD87 is the best source debugger for embedded development and the only product offering high performance and advanced software In Circuit Emulator capabilities. This package and your compiler form a complete solution for the development of embedded programs. Order now and receive a FREE copy of Native DSD87, a $125 value.

Remote DSD87 (supports both Microsoft and Turbo C) . . . $395
Remote DSD87 OMF (includes Intel OMF support) $795

Link Link Link Link Link Link Link Link

ROM-Link is the first integrated link and locate utility to directly generate COM, EXE, SYS, HEX, or Intel OMF files. The only linker ever guaranteed faster than any present or future competitors. Supports Microsoft debugging information. Receive a FREE librarian and cross reference utility with your order!

ROM-Link $395

Soft Advances
10811 Washington Bl., Ste. 205 • Culver City, CA 90232 • (213) 559-7015

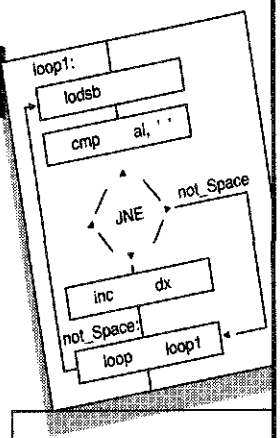
Number **33** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ASMFLOW $99.95

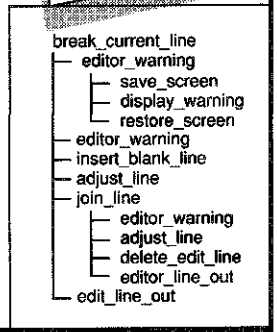
AT LAST!

YOU CAN STEP BACK AND TAKE A LOOK AT YOUR CODE



FLOW CHART AND ANALYZE YOUR ASSEMBLY LANGUAGE SOURCE CODE

- Flow Charts
- Tree Diagrams
- Stack Sizing
- Register Analysis
- CPU Timing Analysis
- Procedural X-Reference
- 8088/87 to 80386/387
- Context-Sensitive Help
- Menu/Batch/Command line Operation
- MASM 5.1 Compatible



QUANTUM SOFTWARE

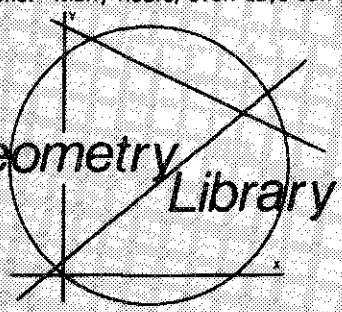
19855 Stevens Creek Blvd, Suite 154
Cupertino, CA 95014
(408) 244-6826 – Free Demo

VISA - MC 30 - Day Money Back Guarantee

Number 252 on the Reader Service Card

*"The cost involved, in writing just one **TurboGeometry** routine, is more than the price of the library itself".*

Are you programming or planning to program in CAD/CAM or graphics applications? Many hours, even days can be spent in writing and debugging geometric routines. ***Turbo-Geometry library*** can relieve you of these time consuming tasks. There are over 150 routines in the library. Some of the areas covered are: Equations of lines, circles, and planes • Intersection of lines, circle, areas and planes • 2 and 3 dimensional transforms • Decomposition of polygon • Hidden line removal • Volume and area calculations • Perspectives • Polygon line clipping • And many more! Supported by example programs, and a 400 page manual. Source code included. 30 day guarantee. Need IBM PC/COMP, MS/PC DOS 2.0+, Turbo C, Turbo Pascal 4.0, MS C or Turbo Pascal MAC. $149.95 plus $5.00 S&H. Foreign orders add $11.00 for S&H. VISA, MC, Chk., MO. To order, call or write.

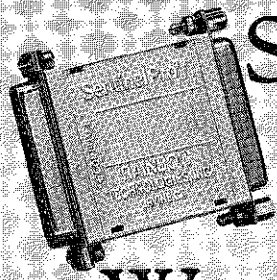


Disk Software, Inc. • 2116 E. Arapaho, #487
Richardson, TX 75081 • Tel #214-423-7288

"Give yourself a break. Get TurboGeometry Today"

Number 229 on the Reader Service Card

Suppose You Protected Your Software & The Silence Was Golden.



Software protected by Rainbow Technologies' Software Sentinel™ has a sound of its own: Golden. And unlike copy-protected software, very silent.

Golden because your development investment is protected. Down to each and every copy you sell. To give you the long-term solution to revenue loss. And the sure way out of the worldwide free-distribution network.

Silent because Software Sentinel protection is transparent. With easy-to-live-with reliability. And no impact on the user or your software's performance. Which means all you hear are the nods of approval.

The Software Sentinel key is shipped with the software and easily installed on the PC's printer port. Protected software will run only when it's installed. No key – No use. It's that simple.

Key Sentinel Family Features.

□ "Execution control" technique prohibits unauthorized use of software □ Permits unlimited backup copies □ Virtually unbreakable □ For DOS, OS/2 and XENIX □ Pocketsize □ Transportable □ Transparent operation □ Use with a hard disk

SentinelPro™ — Sentinel-C

□ Protects multiple packages □ Runs under DOS, OS/2 and Xenix on IBM PC/XT/AT and PS/2 and compatibles □ Supports floating point □ Supports multiple keys □ Allows all non-volatile memory to be used □ Supports Clipper, dBASE, Quicksilver, Turbo Pascal, Turbo C, Microsoft C, Microsoft Pascal, Microsoft Basic, Quick Basic, Basic Compiler, Quick C, Quick Pascal, Microsoft Fortran, Lattice C, Lattice Pascal, Lattice Basic, Lattice Fortran, Lattice C++, Aztec C, Aztec Pascal, Aztec Basic, Aztec Fortran, Aztec C++, Watcom C, Watcom Fortran, Watcom Basic, Watcom Pascal, Watcom C++, Microsoft Macro Assembler, Turbo Assembler, Borland C++, Zortech C++, Oasys C++, Microsoft C++, Microsoft Quick C++, Turbo C++, Microsoft Windows, Windows 386, Windows NT, Windows Pascal, Windows C, Windows Basic, Windows Fortran, Windows C++, Windows Assembler, Windows Debugger, Windows Profiler, Windows Spy, Windows Help Compiler, Windows Resource Compiler, Windows Resource Workshop, Windows Software Development Kit, Windows Application Compiler, Windows Application Interpreter, Windows Application Debugger, Windows Application Profiler, Windows Application Spy, Windows Application Help Compiler, Windows Application Resource Compiler, Windows Application Resource Workshop, Windows Application

POWER GRAPHICS FOR YOUR PC!

MetaWINDOW

NO ROYALTIES!

PC TECH JOURNAL
Product of the Month
"Awarded for excellence in product design and development"

Developing a graphics application? Move to the peak of performance and graphics functionality with MetaWINDOW! Built on state of the art "object oriented" display flexibility, now you can take advantage of powerful advanced graphics capabilities like those used on the Macintosh, Sun and Apollo workstations. Using MetaWINDOW you can break the bottleneck between limited function low-level graphic primitive libraries and slow, bulky hard-coded window managers.

MetaWINDOW provides an advanced set of graphic drawing functions, dynamic runtime support for a full range of graphic peripherals, plus unmatched performance for designing multi-window desktop applications.

Rated #1 by Users!

J. Kayser, CMSI -
"MetaWINDOW is the best graphics package I've seen. Since I bought my PC, I've spent over $2500 on software. I've bought bad software, mediocre software, and good software. There's not much software I call excellent, but MetaWINDOW is excellent!"

David Gray, LUCAS LABS -
"I would also like to take this time to compliment you on providing such a fantastic graphics package. We have used several packages previously including Turbo graphics and Halo graphics and found none that compare to the versatile, high performance graphics package that your company offers. We have also found your technical support staff to be one of the best we have ever dealt with."

John Weaver, Chairperson
Department of Mathematical Sciences
WEST CHESTER UNIVERSITY -
"We have been using MetaWINDOW for our graphics instruction for the past several semesters and are pleased with the package. This past summer we upgraded one system to MetaWINDOW version 3 and were so impressed that we want to upgrade all systems prior to the Spring semester. This is a first class package and I think it is the best for teaching computer graphics."

Awarded Top Honors!

PC Tech Journal, Jul '85
Product of the Month - *"... a technological tour de force for fast PC graphics." "... one high-powered piece of artillery."*

Computer Language, Oct '87 -
*"... surprisingly powerful for the cost." Independently benchmarked the fastest of all graphics systems reviewed - faster than Halo, over 3 times faster than GSS*CGI.*

Turbo Technix, Feb '88
"... Customer support is excellent. I've called Metagraphics several times and have gotten prompt, accurate help each time. Metagraphics has a winner with TurboWINDOW."

Reach For The Future!

MetaWINDOW - $195
Advanced Graphics Development Toolkit
MetaWINDOW comes complete with language bindings for 20 popular C, Pascal and Fortran compilers, plus dynamic runtime support for over 80 graphics adaptors and input devices.
4 disks, 3 260 page manuals

TurboWINDOW/C - $95
TurboWINDOW/Pascal - $95
QuickWINDOW/C - $95
All the features of MetaWINDOW for Borland Turbo C, Turbo Pascal or Microsoft C.

NEW - FontWINDOW - $95
MetaWINDOW font/icon editor plus 100 fonts.

NEW - MetaWINDOW/Premium - $495
Power graphics for new high-res, high performance TI34010 based graphics cards!
(Licensing required for redistribution of MetaWINDOW/Premium)

Superior Functionality!

- Macintosh QuickDraw function superset - *simplifies porting PC & Mac applications.*
- PostScript & X-Window compatible drawing attributes - the defacto graphics *standard for PC X-Window servers.*
- Supports both bitmap and outline fonts - *dynamically face fonts for bold, italic, underline or strike-out effects.*
- Full 16 function "rasterOp" flexibility - *easily write, erase, rubberband or drag: lines, text, icons, bit images and complex objects.*
- Dynamic recognition and run-time support for 80+ popular graphics cards - *eliminates multitudes of clumsy device drivers and bothersome configuration sessions - true device independence at last!*
- Virtual images - *create and manipulate graphic images in local or expanded memory up to 8 megabytes in size.*
- *... and the list goes on!*

30 Day Money Back Guarantee
ORDER TODAY! 800-332-1550
For information or in CA call 408-438-1550

METAGRAPHICS
SOFTWARE CORPORATION
269 Mount Herman Road
Scotts Valley, CA 95066

Number 143 on the Reader Service Card

Two 256-Color Pages

Listing 2 demonstrates the two pages of 320x400 256-color mode by drawing slanting color bars in page 0, drawing color bars slanting the other way in page 1, and flipping to page 1 on the next key press. (Note that page 1 is accessed starting at offset 8000h in display memory, and is—not surprisingly—displayed by setting the start address to 8000h.) Finally, Listing 2 draws vertical color bars in page 0 and flips back to page 0 when another key is pressed.

The color bar routines don't use the WritePixel subroutine from Listing 1; they go straight to display memory instead for improved speed. As I mentioned above, yet better speed could be achieved by a color bar algorithm that draws all the pixels in plane 0, then all the pixels in plane 1, and so on, thereby avoiding the overhead of constantly reprogramming the Map Mask register.

When you run Listing 2, note the extremely smooth edges and fine gradations of color, especially in the screens with slanting color bars. The displays produced by Listing 2 make it clear that 320x400 256-color mode can produce effects that are simply not possible in the 16-color modes.

Something to Think About

You can, if you wish, use the display-memory organization of 320x400 mode in 320x200 mode by modifying Set320x400Mode to leave the maximum scan line setting at 1 in the mode set. (The version of Set320x400Mode in listings 1 and 2 forces the maximum scan line to 0, doubling the effective resolution of the screen.) Why would you want to do that? Simply because you could then choose from, not two, but *four* 320x200 256-color display pages, starting at offsets 0, 4000h, 8000h, and 0C000h in display memory.

In Conclusion

Who knows what resolutions lurk in the hearts of VGAs? Now you do. What an adapter, eh?

As always, I'm not sure what's next on the list for "On Graphics." Also as always, I'm open to your suggestions; drop me a line via MCI or the U.S. mail.

See you next time. ♦

Michael Abrash is a Senior Software Engineer for Orion Instruments, Inc., of Redwood City, California, where he is working on the design of advanced PC-based test instrumentation. Readers may write to Michael at P.O. Box 39051, Mountain View, CA 94039, or drop Michael a line via MCI Mail (user name MABRASH).

Code follows.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

All code in this issue is available on 360K, MS-DOS disk.
ORDER BY PHONE — VISA/MC orders call toll free 800/234-0386.
Mail order information on pages 8 and 88.

Code from "256-Color Resolution" by Michael Abrash

1

```
;
; *** Listing 1 ***
;
; Program to demonstrate pixel drawing in 320x400 256-color
; mode on the VGA. Draws 8 lines to form an octagon, a pixel
; at a time. Draws 8 octagons in all, one on top of the other,
; each in a different color set. Although it's not used, a
; pixel read function is also provided.
;
VGA_SEGMENT     equ     0a000h
SC_INDEX        equ     3c4h    ;Sequence Controller Index register
GC_INDEX        equ     3ceh    ;Graphics Controller Index register
CRTC_INDEX      equ     3d4h    ;CRT Controller Index register
MAP_MASK        equ     2       ;Map Mask register index in SC
MEMORY_MODE     equ     4       ;Memory Mode register index in SC
MAX_SCAN_LINE   equ     9       ;Maximum Scan Line reg index in CRTC
START_ADDRESS_HIGH equ  0ch     ;Start Address High reg index in CRTC
UNDERLINE       equ     14h     ;Underline Location reg index in CRTC
MODE_CONTROL    equ     17h     ;Mode Control register index in CRTC
READ_MAP        equ     4       ;Read Map register index in GC
GRAPHICS_MODE   equ     5       ;Graphics Mode register index in GC
MISCELLANEOUS   equ     6       ;Miscellaneous register index in GC
SCREEN_WIDTH    equ     320     ;# of pixels across screen
SCREEN_HEIGHT   equ     400     ;# of scan lines on screen
WORD_OUTS_OK    equ     1       ;set to 0 to assemble for
                              ; computers that can't handle
                              ; word outs to indexed VGA registers
;
stack   segment para stack 'STACK'
        db      512 dup (?)
stack   ends
;
Data    segment word 'DATA'
;
BaseColor       db      0
;
; Structure used to control drawing of a line.
;
LineControl     struc
StartX          dw      ?
StartY          dw      ?
LineXInc        dw      ?
LineYInc        dw      ?
BaseLength      dw      ?
LineColor       db      ?
LineControl     ends
;
; List of descriptors for lines to draw.
;
LineList        label   LineControl
        LineControl     <130,110,1,0,60,0>
        LineControl     <190,110,1,1,60,1>
        LineControl     <250,170,0,1,60,2>
        LineControl     <250,230,-1,1,60,3>
        LineControl     <190,290,-1,0,60,4>
        LineControl     <130,290,-1,-1,60,5>
        LineControl     <70,230,0,-1,60,6>
        LineControl     <70,170,1,-1,60,7>
        LineControl     <-1,0,0,0,0,0>
Data    ends
;
; Macro to output a word value to a port.
;
OUT_WORD        macro
if WORD_OUTS_OK
        out     dx,ax
else
        out     dx,al
        inc     dx
        xchg    ah,al
        out     dx,al
        dec     dx
        xchg    ah,al
endif
        endm
;
; Macro to output a constant value to an indexed VGA register.
;
CONSTANT_TO_INDEXED_REGISTER    macro   ADDRESS, INDEX, VALUE
        mov     dx,ADDRESS
        mov     ax,(VALUE shl 8) + INDEX
        OUT_WORD
        endm
;
Code    segment
        assume  cs:Code, ds:Data
Start   proc    near
        mov     ax,Data
        mov     ds,ax
;
;Set 320x400 256-color mode.
;
        call    Set320x400Mode
;
; We're in 320x400 256-color mode. Draw each line in turn.
;
ColorLoop:
        mov     si,offset LineList ;point to the start of the
                                   ; line descriptor list
LineLoop:
        mov     cx,[si+StartX]  ;set the initial X coordinate
        cmp     cx,-1
        jz      LinesDone       ;a descriptor with a -1 X
                                ; coordinate marks the end
                                ; of the list
        mov     dx,[si+StartY]     ;set the initial Y coordinate,
        mov     bl,[si+LineColor]  ; line color,
        mov     bp,[si+BaseLength] ; and pixel count
        add     bl,[BaseColor]  ;adjust the line color according
                                ; to BaseColor
PixelLoop:
        push    cx              ;save the coordinates
        push    dx
        call    WritePixel      ;draw this pixel
        pop     dx              ;retrieve the coordinates
        pop     cx
        add     cx,[si+LineXInc] ;set the coordinates of the
        add     dx,[si+LineYInc] ; next point of the line
        dec     bp              ;any more points?
        jnz     PixelLoop       ;yes, draw the next
        add     si,size LineControl ;point to the next line descriptor
        jmp     LineLoop        ; and draw the next line
LinesDone:
        call    GetNextKey      ;wait for a key, then
        inc     [BaseColor]     ; bump the color selection and
        cmp     [BaseColor],8   ; see if we're done
        jb      ColorLoop       ;not done yet
;
; Wait for a key and return to text mode and end when
; one is pressed.
;
        call    GetNextKey
        mov     ax,0003h
        int     10h     ;text mode
        mov     ah,4ch
        int     21h     ;done
;
Start   endp
;
; Sets up 320x400 256-color modes.
;
; Input: none
;
; Output: none
;
Set320x400Mode  proc    near
;
; First, go to normal 320x200 256-color mode, which is really a
; 320x400 256-color mode with each line scanned twice.
;
        mov     ax,0013h  ;AH = 0 means mode set, AL = 13h selects
```

Code continues . . .

2

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Drawbridge

Drawbridge™ is an interactive graphics editor that lets you create high-quality graphic displays for your programs, but saves you the tedious task of programming them.

That's right! Just draw a picture on the screen and Drawbridge will generate a file of source code calls to your graphics library at the touch of a key. Copy the code into your program, compile, link, and that's it.

- Object-oriented editing
- Increases productivity
- Source code generation for specific graphics library/language
- Uses library features such as color, line styles, fill patterns, etc.

Separate versions of Drawbridge are available to generate source code calls to these popular graphics libraries:

- MetaWINDOW† (C or Pascal)
- Microsoft Graphics Library for MSC V5 and QuickC
- Borland Graphics Interface for TurboC and TurboPascal

Drawbridge comes with a 30-day money back guarantee.†† Order your copy today! To order by phone or to get more information call:

(217) 359-1878

"Excellent Product!! Most useful utility I've had in years."
R.B., San Diego, CA

"Keep up the good work!"
P.N., Portland, OR

"Works very nice—EXCELLENT Product!"
R.S., St. Louis, MO

Courseware Applications, Inc.
475 Devonshire Drive
Champaign, IL 61820

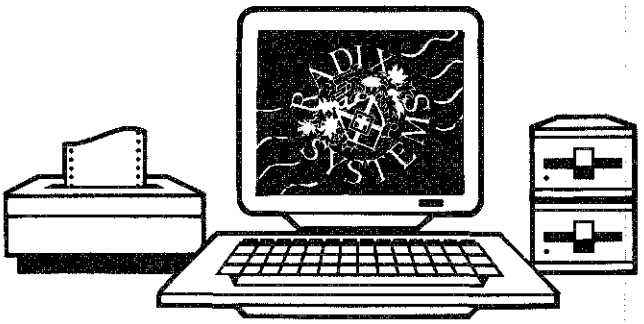
Drawbridge for:

MetaWINDOW	$129.00
Microsoft Graphics Library	$49.00
Borland Graphics Interface	$49.00

† MetaWINDOW is a trademark of Metagraphics Software Corporation. QuickC is a trademark of Microsoft Corp. TurboC and TurboPascal are trademarks of Borland International.

†† A small restocking fee applies to all returns.

Number **250** on the Reader Service Card



BRING DATABASE POWER TO YOUR SCREEN

Now you can achieve database flexibility in screen processing with the Radix Screen Management System. It's a language independent screen and keyboard controller. The package includes a facility that allows full screen editing of screen maps. Once created, a screen map is allocated and released at runtime, no compiling or linking necessary. Use the screen map in different applications developed under different languages. There are only eight commands to learn which control the screen maps. In addition, an Internal Document Facility will automatically display and scroll a window of your application's help text.

Other features include:

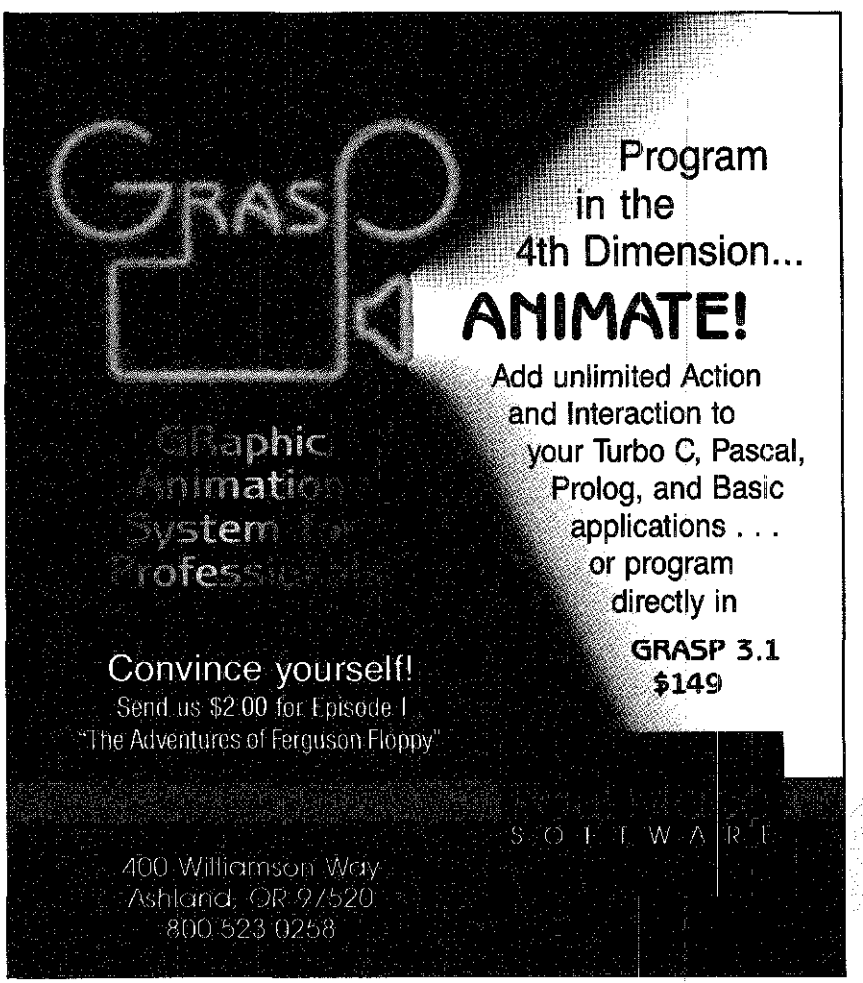
- **Window and Overlay Screens**
- **Alter Attributes at Runtime**
- **Move Data Field-by-Field or in groups**
- **Request Only Modified Values**
- **Update the Screen Map Anytime**
- **Extensive Documentation**
- **Uses Standard Command Interface**

To order, send $198 plus $7 for shipping. No royalties.

Requires an IBM PC or compatible, 512KB of memory, and DOS 2.0 or higher. UNIX and OS/2 versions will be available soon. For more information call (201) 697-2630, or send $5 for demonstration diskette and Concepts & Facilities manual.

RADIX SYSTEMS INC.
P.O. Box 306
Rochelle Park, NJ 07662

Number **251** on the Reader Service Card



Number **80** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

3

```
                        ; 256-color graphics mode
        int     10h     ;BIOS video interrupt
;
; Change CPU addressing of video memory to linear (not odd/even,
; chain, or chain 4), to allow us to access all 256K of display
; memory. When this is done, VGA memory will look just like memory
; in modes 10h and 12h, except that each byte of display memory will
; control one 256-color pixel, with 4 adjacent pixels at any given
; address, one pixel per plane.
;
        mov     dx,SC_INDEX
        mov     al,MEMORY_MODE
        out     dx,al
        inc     dx
        in      al,dx
        and     al,not 08h      ;turn off chain 4
        or      al,04h          ;turn off odd/even
        out     dx,al
        mov     dx,GC_INDEX
        mov     al,GRAPHICS_MODE
        out     dx,al
        inc     dx
        in      al,dx
        and     al,not 10h      ;turn off odd/even
        out     dx,al
        dec     dx
        mov     al,MISCELLANEOUS
        out     dx,al
        inc     dx
        in      al,dx
        and     al,not 02h      ;turn off chain
        out     dx,al
;
; Now clear the whole screen, since the mode 13h mode set only
; cleared 64K out of the 256K of display memory. Do this before
; we switch the CRTC out of mode 13h, so we don't see garbage
; on the screen when we make the switch.
;
        CONSTANT_TO_INDEXED_REGISTER SC_INDEX,MAP_MASK,0fh
                                ;enable writes to all planes, so
                                ; we can clear 4 pixels at a time
        mov     ax,VGA_SEGMENT
        mov     es,ax
        sub     di,di
        mov     ax,di
        mov     cx,8000h        ;# of words in 64K
        cld
        rep     stosw           ;clear all of display memory
;
; Tweak the mode to 320x400 256-color mode by not scanning each
; line twice.
;
        mov     dx,CRTC_INDEX
        mov     al,MAX_SCAN_LINE
        out     dx,al
        inc     dx
        in      al,dx
        and     al,not 1fh      ;set maximum scan line = 0
        out     dx,al
        dec     dx
;
; Change CRTC scanning from doubleword mode to byte mode, allowing
; the CRTC to scan more than 64K of video data.
;
        mov     al,UNDERLINE
        out     dx,al
        inc     dx
        in      al,dx
        and     al,not 40h      ;turn off doubleword
        out     dx,al
        dec     dx
        mov     al,MODE_CONTROL
        out     dx,al
        inc     dx
        in      al,dx
        or      al,40h  ;turn on the byte mode bit, so memory is
                        ; scanned for video data in a purely
                        ; linear way, just as in modes 10h and 12h
        out     dx,al
```

4

```
        ret
Set320x400Mode endp
;
; Draws a pixel in the specified color at the specified
; location in 320x400 256-color mode.
;
; Input:
;       CX = X coordinate of pixel
;       DX = Y coordinate of pixel
;       BL = pixel color
;
; Output: none
;
; Registers altered: AX, CX, DX, DI, ES
;
WritePixel      proc    near
        mov     ax,VGA_SEGMENT
        mov     es,ax ;point to display memory
        mov     ax,SCREEN_WIDTH/4
                        ;there are 4 pixels at each address, so
                        ; each 320-pixel row is 80 bytes wide
                        ; in each plane
        mul     dx      ;point to start of desired row
        push    cx      ;set aside the X coordinate
        shr     cx,1    ;there are 4 pixels at each address
        shr     cx,1    ; so divide the X coordinate by 4
        add     ax,cx   ;point to the pixel's address
        mov     di,ax
        pop     cx      ;get back the X coordinate
        and     cl,3    ;get the plane # of the pixel
        mov     ah,1
        shl     ah,cl   ;set the bit corresponding to the plane
                        ; the pixel is in
        mov     al,MAP_MASK
        mov     dx,SC_INDEX
        OUT_WORD        ;set to write to the proper plane for
                        ; the pixel
        mov     es:[di],bl      ;draw the pixel
        ret
WritePixel      endp
;
; Reads the color of the pixel at the specified location in 320x400
; 256-color mode.
;
; Input:
;       CX = X coordinate of pixel to read
;       DX = Y coordinate of pixel to read
;
; Output:
;       AL = pixel color
;
; Registers altered: AX, CX, DX, SI, ES
;
ReadPixel       proc    near
        mov     ax,VGA_SEGMENT
        mov     es,ax   ;point to display memory
        mov     ax,SCREEN_WIDTH/4
                        ;there are 4 pixels at each address, so
                        ; each 320-pixel row is 80 bytes wide
                        ; in each plane
        mul     dx      ;point to start of desired row
        push    cx      ;set aside the X coordinate
        shr     cx,1    ;there are 4 pixels at each address
        shr     cx,1    ; so divide the X coordinate by 4
        add     ax,cx   ;point to the pixel's address
        mov     si,ax
        pop     ax      ;get back the X coordinate
        and     al,3    ;get the plane # of the pixel
        mov     ah,al
        mov     al,READ_MAP
        mov     dx,GC_INDEX
        OUT_WORD        ;set to read from the proper plane for
                        ; the pixel
        lods    byte ptr es:[si] ;read the pixel
        ret
ReadPixel       endp
;
; Waits for the next key and returns it in AX.
;
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Abrash Code, Continued

5

```
; Input: none
;
; Output:
;       AX = full 16-bit code for key pressed
;
GetNextKey      proc    near
WaitKey:
        mov     ah,1
        int     16h
        jz      WaitKey ;wait for a key to become available
        sub     ah,ah
        int     16h     ;read the key
        ret
GetNextKey      endp
;
Code    ends
;
        end     Start


;
; *** Listing 2 ***
;
; Program to demonstrate the two pages available in 320x400
; 256-color modes on a VGA.  Draws diagonal color bars in all
; 256 colors in page 0, then does the same in page 1 (but with
; the bars tilted the other way), and finally draws vertical
; color bars in page 0.
;
VGA_SEGMENT     equ     0a000h
SC_INDEX        equ     3c4h    ;Sequence Controller Index register
GC_INDEX        equ     3ceh    ;Graphics Controller Index register
CRTC_INDEX      equ     3d4h    ;CRT Controller Index register
MAP_MASK        equ     2       ;Map Mask register index in SC
MEMORY_MODE     equ     4       ;Memory Mode register index in SC
MAX_SCAN_LINE   equ     9       ;Maximum Scan Line reg index in CRTC
START_ADDRESS_HIGH equ  0ch     ;Start Address High reg index in CRTC
UNDERLINE       equ     14h     ;Underline Location reg index in CRTC
MODE_CONTROL    equ     17h     ;Mode Control register index in CRTC
GRAPHICS_MODE   equ     5       ;Graphics Mode register index in GC
MISCELLANEOUS   equ     6       ;Miscellaneous register index in GC
SCREEN_WIDTH    equ     320     ;# of pixels across screen
SCREEN_HEIGHT   equ     400     ;# of scan lines on screen
WORD_OUTS_OK    equ     1       ;set to 0 to assemble for
                                ; computers that can't handle
                                ; word outs to indexed VGA registers
;
stack   segment para stack 'STACK'
        db      512 dup (?)
stack ends
;
; Macro to output a word value to a port.
;
OUT_WORD        macro
if WORD_OUTS_OK
        out     dx,ax
else
        out     dx,al
        inc     dx
        xchg    ah,al
        out     dx,al
        dec     dx
        xchg    ah,al
endif
        endm
;
; Macro to output a constant value to an indexed VGA register.
;
CONSTANT_TO_INDEXED_REGISTER    macro   ADDRESS, INDEX, VALUE
        mov     dx,ADDRESS
        mov     ax,(VALUE shl 8) + INDEX
        OUT_WORD
        endm
;
Code    segment
        assume  cs:Code
Start   proc    near
;
; Set 320x400 256-color mode.
```

6

```
;           call    Set320x400Mode
;
; We're in 320x400 256-color mode, with page 0 displayed.
; Let's fill page 0 with color bars slanting down and to the right.
;
        sub     di,di           ;page 0 starts at address 0
        mov     bl,1            ;make color bars slant down and
                                ; to the right
        call    ColorBarsUp     ;draw the color bars
;
; Now do the same for page 1, but with the color bars
; tilting the other way.
;
        mov     di,8000h        ;page 1 starts at address 8000h
        mov     bl,-1           ;make color bars slant down and
                                ; to the left
        call    ColorBarsUp     ;draw the color bars
;
; Wait for a key and flip to page 1 when one is pressed.
;
        call    GetNextKey
        CONSTANT_TO_INDEXED_REGISTER CRTC_INDEX,START_ADDRESS_HIGH,80h
                                ;set the Start Address High register
                                ; to 80h, for a start address of 8000h
;
; Draw vertical bars in page 0 while page 1 is displayed.
;
        sub     di,di           ;page 0 starts at address 0
        sub     bl,bl           ;make color bars vertical
        call    ColorBarsUp     ;draw the color bars
;
; Wait for another key and flip back to page 0 when one is pressed.
;
        call    GetNextKey
        CONSTANT_TO_INDEXED_REGISTER CRTC_INDEX,START_ADDRESS_HIGH,00h
                                ;set the Start Address High register
                                ; to 00h, for a start address of 0000h
;
; Wait for yet another key and return to text mode and end when
; one is pressed.
;
        call    GetNextKey
        mov     ax,0003h
        int     10h     ;text mode
        mov     ah,4ch
        int     21h     ;done
;
Start   endp
;
; Sets up 320x400 256-color modes.
;
; Input: none
;
; Output: none
;
Set320x400Mode  proc    near
;
; First, go to normal 320x200 256-color mode, which is really a
; 320x400 256-color mode with each line scanned twice.
;
        mov     ax,0013h  ;AH = 0 means mode set, AL = 13h selects
                          ; 256-color graphics mode
        int     10h     ;BIOS video interrupt
;
; Change CPU addressing of video memory to linear (not odd/even,
; chain, or chain 4), to allow us to access all 256K of display
; memory. When this is done, VGA memory will look just like memory
; in modes 10h and 12h, except that each byte of display memory will
; control one 256-color pixel, with 4 adjacent pixels at any given
; address, one pixel per plane.
;
        mov     dx,SC_INDEX
        mov     al,MEMORY_MODE
        out     dx,al
        inc     dx
        in      al,dx
        and     al,not 08h      ;turn off chain 4
        or      al,04h          ;turn off odd/even
        out     dx,al
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

7

```
        mov     dx,GC_INDEX
        mov     al,GRAPHICS_MODE
        out     dx,al
        inc     dx
        in      al,dx
        and     al,not 10h      ;turn off odd/even
        out     dx,al
        dec     dx
        mov     al,MISCELLANEOUS
        out     dx,al
        inc     dx
        in      al,dx
        and     al,not 02h      ;turn off chain
        out     dx,al

;
; Now clear the whole screen, since the mode 13h mode set only
; cleared 64K out of the 256K of display memory. Do this before
; we switch the CRTC out of mode 13h, so we don't see garbage
; on the screen when we make the switch.
;
        CONSTANT_TO_INDEXED_REGISTER SC_INDEX,MAP_MASK,0Fh
                                        ;enable writes to all planes, so
                                        ; we can clear 4 pixels at a time
        mov     ax,VGA_SEGMENT
        mov     es,ax
        sub     di,di
        mov     ax,di
        mov     cx,8000h        ;# of words in 64K
        cld
        rep     stosw           ;clear all of display memory

;
; Tweak the mode to 320x400 256-color mode by not scanning each
; line twice.
;
        mov     dx,CRTC_INDEX
        mov     al,MAX_SCAN_LINE
        out     dx,al
        inc     dx
        in      al,dx
        and     al,not 1fh      ;set maximum scan line = 0
        out     dx,al
        dec     dx

;
; Change CRTC scanning from doubleword mode to byte mode, allowing
; the CRTC to scan more than 64K of video data.
;
        mov     al,UNDERLINE
        out     dx,al
        inc     dx
        in      al,dx
        and     al,not 40h      ;turn off doubleword
        out     dx,al
        dec     dx
        mov     al,MODE_CONTROL
        out     dx,al
        inc     dx
        in      al,dx
        or      al,40h  ;turn on the byte mode bit, so memory is
                        ; scanned for video data in a purely
                        ; linear way, just as in modes 10h and 12h
        out     dx,al
        ret
Set320x400Mode  endp
;
; Draws a full screen of slanting color bars in the specified page.
;
; Input:
;       DI = page start address
;       DL = 1 to make the bars slant down and to the right, -1 to
;            make them slant down and to the left, 0 to make
;            them vertical.
;
ColorBarsUp     proc    near
        mov     ax,VGA_SEGMENT
        mov     es,ax   ;point to display memory
        sub     bh,bh   ;start with color 0
        mov     si,SCREEN_HEIGHT ;# of rows to do
        mov     dx,SC_INDEX
        mov     al,MAP_MASK
        mov     ah,bh   ;start with color 0
```

Code continues . . .

SCREEN MANAGER PROFESSIONAL

ADVANCED WINDOWING

• The Screen Manager Professional gives your applications the finest bullet proof windowing interface available. All functions have been carefully designed and tested with the programmer in mind. The functions are easy to use and remember. Although other products may boast of hundreds of functions, they often involve needless repetition. Our functions are elegant and powerful. Each accomplishes a variety of tasks with one command. All functions are written in hand-optimized assembly language to provide the fastest possible response and smallest possible size.

ORDER

• To receive your copy of the **Screen Manager Professional, send $139.95.** VISA, MasterCard, Check, or Money Order accepted. Sorry, no C.O.D.'s. Please allow 2-4 weeks for delivery. PA residents add 6% sales tax.

IMPORTANT FEATURES

- Unlimited number of windows
- Complete writing and scrolling support for overlapping windows
- VGA, EGA, CGA, MGA support
- Write to video ram or use BIOS
- Multiple monitor support
- No fuzz or flicker
- Drag windows around the screen
- Menu system
- Resize and reposition windows
- Save and Load windows from disk
- MS-DOS memory allocation
- 100% hand-optimized assembly
- Over 90 powerful functions
- Royalty free!

FOR USE WITH

• Datalight C • Microsoft C
• Microsoft Quick C • Aztec C • Turbo C
• Microsoft Quickbasic • Turbo Pascal

EASY TO USE

• Free yourself to concentrate on the substance of your program. Over 90 functions provide all the windowing tools needed to develop serious applications whether you are a professional or a hobbyist.

• The Screen Manager Professional is a powerful windowing facility. You can increase your productivity as well as the overall quality of your work. The most important part of any program is its ease of use. The Screen Manager Professional can help you create easy to use, visually stimulating, programs with a minimum of effort. Now powerful windowing is available to you at an affordable price. If you have further questions, please call or write.

Programming Utilities for Professionals

LAI

LOGICAL ALTERNATIVES INCORPORATED
Calder Square
P.O. Box 10674
State College, PA 16805-0674
(814) 234-8088

Number 246 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Abrash Code, Continued

```
        out     dx,al   ;point the SC Index reg to the Map Mask reg
        inc     dx      ;point DX to the SC Data register
RowLoop:
        mov     cx,SCREEN_WIDTH/4
                        ;there are 4 pixels at each address, so
                        ; each 320-pixel row is 80 bytes wide
                        ; in each plane
        push    bx      ;save the row-start color
ColumnLoop:
MAP_SELECT = 1
        rept    4       ;do all 4 pixels at this address with
                        ; in-line code
        mov     al,MAP_SELECT
        out     dx,al   ;select planes 0, 1, 2, and 3 in turn
        mov     es:[di],bh ;write this plane's pixel
        inc     bh      ;set the color for the next pixel
MAP_SELECT = MAP_SELECT shl 1
        endm
        inc     di      ;point to the address containing the next
                        ; 4 pixels
        loop    ColumnLoop ;do any remaining pixels on this line
        pop     bx      ;get back the row-start color
        add     bh,bl   ;select next row-start color (controls
                        ; slanting of color bars)
        dec     si      ;count down lines on the screen
        jnz     RowLoop
        ret
ColorBarsUp     endp
;
; Waits for the next key and returns it in AX.
;
GetNextKey      proc    near
WaitKey:
        mov     ah,1
        int     16h
        jz      WaitKey ;wait for a key to become available
        sub     ah,ah
        int     16h     ;read the key
        ret
GetNextKey      endp
;
Code    ends
;
        end     Start
```


Desktop 9-Track Tape Subsystem

Now, 9-track tape lets your micro exchange data with minis and mainframes.

Simply exchange data files on a reel of 9-track tape.

9-TRACK is the first choice for file interchange among data processing professionals. Now, Qualstar's low cost 1/2-inch 9-track *Ministreamer* tape systems bring full ANSI data interchange to IBM PCs or Macintosh, giving your micro the freedom to exchange data files with nearly any mainframe or minicomputer in the world.

Available in both 7" and 10-1/2" versions, compact Qualstar tape drives can sit on your desktop, *using less space than an ordinary sheet of paper*. Systems include DOS or XENIX compatible software, coupler card and cables. High reliability 1600 or 6250 BPI capability may be used for disk backup as well as data interchange. Discover the big advantage 9-track tape has over other micro/mainframe links.

DEC HP ALTOS NCR — IBM MAINFRAMES — AT&T UNISYS APOLLO SUN

Call us today!

FOR DETAILS AND TO ORDER:
FAX (818) 882-4081
PHONE (818) 882-5822

#1 Selling 9-Track Systems on the Desktop

QUALSTAR®

9621 Irondale Ave., Chatsworth, CA 91311

©1988 Qualstar Corp. All product and company names and trademarks are the exclusive property of their respective owners.

Number **97** on the Reader Service Card

EVERTRAK

SECURITY

If you HATE Copy Protection, but still want to protect your program from hackers and reverse engineering, EVERTRAK allows you to:

- Shut down DEBUG systems and disassemblers.
- Restrict use by date-range and media-type.
- Embed a secure serial number string.
- Track unauthorized copies back to their source.

EVERTRAK is totally invisible to your end user and uses the latest encryption technology.

EVERTRAK supports IBM & Compatibles using DOS 2.0 or later, sells for $295.00 and has a 30 day money back guarantee.

Az-Tech Software, Inc.
305 East Franklin
Richmond, MO 64085

(800) 227-0644 **(816) 776-2700**

Number 51 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

GRAPHICS TOOLS FOR SOFTWARE DEVELOPERS

What is HALO?

ALO is a device independent library
190 graphics subroutines. It is
mpatible with 17 programming
nguages and over 110 graphics
ardware devices for the IBM PC,
S/2 and compatibles. It provides the
ftware designer with the richest
vironment of graphics functions; the
ogrammer with reliable and well-
ocumented tools; and DP managers
ith continuity of user interface and
atabase format.

Who uses HALO?

nce its introduction in 1982, HALO
as developed an installed base of
,000+ end-users, hundreds of site-
censed corporations, government
gencies, universities, and national
boratories, and most importantly,
family of over 150 Independent Soft-
are Vendors (ISVs) who market ap-
ications written with HALO.

What about performance?

Our ISVs are the best proof of HALO's performance. They use HALO because:

—building their own library costs too much

—HALO outperforms the rest

—HALO is easier to program

—it supports a wider range of devices for a bigger potential market

—new device drivers are added quarterly to prevent software obsolescence and reduce maintenance costs.

HALO™
infinite possibilities in graphics and imaging

How much does it cost?

A single license for HALO costs $300.00 and includes all device drivers and your choice of one compiler binding.

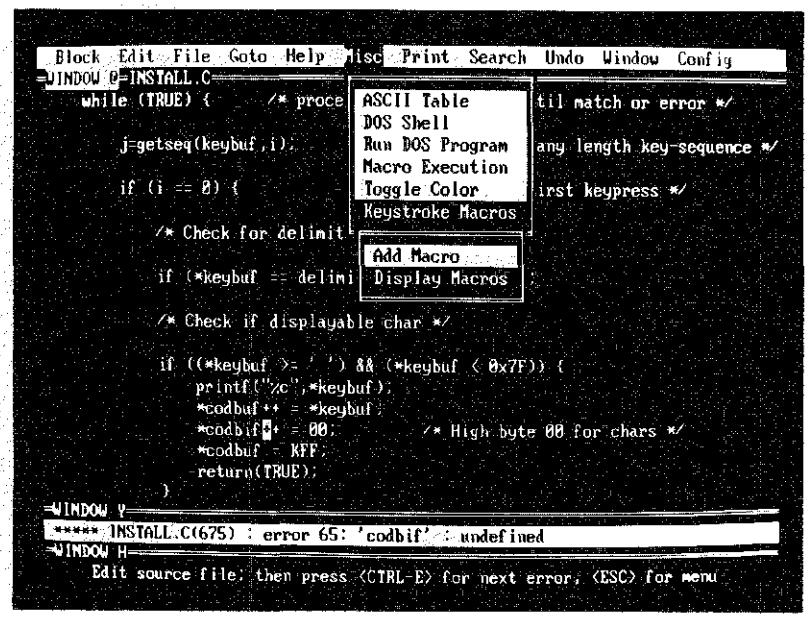
HALO is fully documented, includes LearnHALO (an interactive tutorial) and free 800# technical support.

If you need high performance graphic development software that provides a migration path to OS/2 and other future technology, follow the industry leaders—call **(800) 426-HALO.**

media cybernetics
8484 Georgia Ave.
Silver Spring, MD 20910
(800) 426-HALO
(301) 495-3305 telex 322014

HALO is a registered trademark of Media Cybernetics, Inc.
IBM PC, and PS/2 are registered trademarks of Internatio
Business Machines Corp.
Number **247** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED


Block Edit File Goto Help Misc Print Search Undo Window Config
WINDOW 0 INSTALL.C
while (TRUE) { /* proce til match or error */
j=getseq(keybuf,i); any length key-sequence */
if (i == 0) { irst keypress */
/* Check for delimit
if (*keybuf == delimi
/* Check if displayable char */
if ((*keybuf >= ' ') && (*keybuf < 0x7F)) {
printf("%c",*keybuf);
*codbuf++ = *keybuf;
codbuf++ = 00; / High byte 00 for chars */
*codbuf = KFF;
return(TRUE);
}
ASCII Table
DOS Shell
Run DOS Program
Macro Execution
Toggle Color
Keystroke Macros
Add Macro
Display Macros
WINDOW 9
***** INSTALL.C(675) : error 65: 'codbif' : undefined
WINDOW H
Edit source file, then press <CTRL-E> for next error, <ESC> for menu

VEDIT PLUS

#1 PROGRAMMABLE EDITOR

NEW VERSION 3.0

- **Best Multi-Level Undo**
- **Regular Expressions**
- **Pop-Up ASCII Table**
- **Pull-Down Menus**
- **Compiler Support**
- **Column Blocks**

Until now, if you wanted the best Undo, the best compiler support, regular expressions and column blocks you chose BRIEF™. If you wanted unlimited keystroke macros, the best configurability, "off the cuff" command language macros and blazing speed, you chose VEDIT PLUS.®

Now the Choice is Easy

The all new VEDIT PLUS 3.0 gives you the best Undo of any editor, the best compiler support, unequaled windows, true regular expressions and extensive new features. We're leading the way with easy to use pull down menus, context sensitive help, a pop-up ASCII table, new printing options and much more. Incredibly, VEDIT PLUS 3.0 is now twice as fast as before and, at only 60K in size, it loads fast!

Completely Configurable

Change a few keys or redefine the entire keyboard, VEDIT PLUS adjusts to your editing style in minutes. You can even create new editing functions using simple keystroke macros or fine tune existing ones. VEDIT PLUS is so configurable that it easily emulates other editors and word processors (WordStar and Word Perfect emulation included). Quickly access editing functions with a single key or through the pull-down menus.

Try before You Buy

We challenge you to experience the dazzling performance and exceptional features that make VEDIT PLUS the best choice. Our evaluation disk includes the complete editor.* Learn VEDIT PLUS using our extensive "training" macro that gives instructions in one window while you experiment in another. See for yourself why no other macro language comes close.

Call for your free evaluation copy today. See why VEDIT PLUS has been the #1 choice of programmers, writers and engineers since 1980.

Supports the IBM PC, XT, AT and PS/2 including DESQview, Microsoft Windows, PC-MOS/386, Concurrent DOS and most networks. Also available for MS-DOS, CP/M-86 and FlexOS. (Yes! We support windows on CRT terminals.) $185.

FREE EVALUATION COPY* Call 1-800-45-VEDIT

- **Fully Network Compatible**
- **Call for XENIX and OS/2 versions**
- **30 Day Money-back guarantee**

Features of VEDIT PLUS 3.0

- Simultaneously edit up to 37 files of unlimited size.
- Variable sized windows; multiple windows per file.
- Execute DOS commands and other programs.
- Flexible "cut and paste" with 36 "scratch-pad" buffers.
- Block operations by line, character or column.
- Search with pattern matching or regular expressions.
- Configuration—determine your own keyboard layout, create your own editing functions, support any screen size.
- Select window colors, support 43 line EGA, 50 line VGA.

EASY TO USE

- Modern pull-down menu system. Pop-up ASCII table.
- Context sensitive on-line help is user changeable.
- Multi-level Undo (100 to 1000 levels). Undo keystroke by keystroke or line by line.
- On-line integer calculator (also algebraic expressions).
- Keystroke macros speed editing, menu function "hot keys."

FOR PROGRAMMERS

- Automatic Indent/Undent for "C," PL/I, PASCAL.
- Match/check nested parentheses, e.g. "(" and ")" for "C."
- Flexible macro runs popular compilers and automatically moves cursor to each error in your program. Easily changed to support new compilers and assemblers.

FOR WRITERS

- Word wrap, paragraph formatting and justification.
- Convert to/from Wordstar and mainframe files.
- Flexible printing; fully adjustable margins and Tab stops.

MACRO PROGRAMMING LANGUAGE

- If-then-else, looping, testing, string compare, branching, user prompts, keyboard input, 24 bit algebraic expressions.
- Flexible windowing—forms entry, select size, color, etc.
- Simplifies complex text processing, formatting, conversions.
- Extensive 400 page manual with hundreds of examples.

VEDIT and CompuView are registered trademarks of CompuView Products, Inc. BRIEF is a trademark of UnderWare, Inc. PC-MOS/386 is a trademark of The Software Link, Inc. CP/M-86 and FlexOS are trademarks of Digital Research. MS-DOS, OS/2 and XENIX are trademarks of Microsoft. DESQview is a trademark of Quarterdeck Office Systems.

*Also available for TI Professional, Tandy 2000, DEC Rainbow, WYSE 700, Amdek 1280 and Others.
*Free evaluation disk is fully functional and can even edit small files.

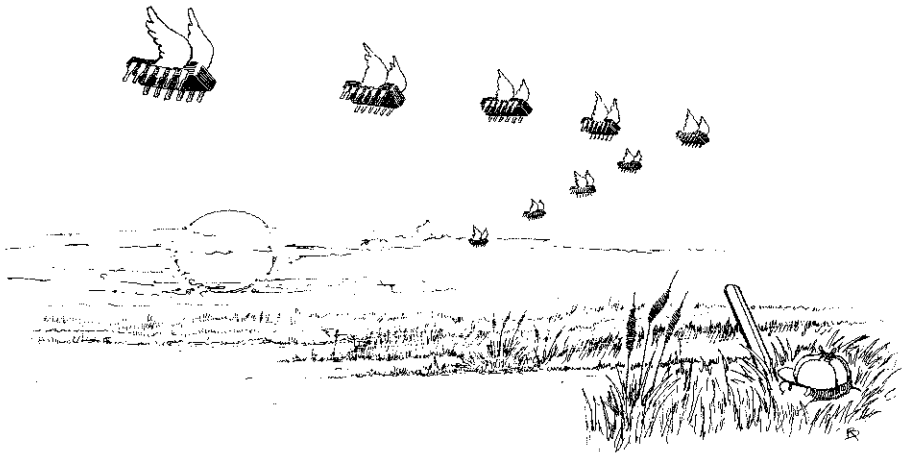
Number 56 on the Reader Service Card

1955 Pauline Blvd., Ann Arbor, MI 48103
(313) 996-1299 • Telex 701821 • Fax (313) 996-1308

CompuView

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Migrating to Graphics Processors



by
Jake Richter

Graphics processors are slowly infiltrating the world around us. They can be found in laser printers, plotters, computer terminals, video games, and, of course, on graphics boards.

With the recent announcement of the Texas Instruments TMS 34020 Graphics System Processor, a second-generation graphics processor, these pieces of silicon have come of age. This indication of TI's commitment to affordable, high-speed graphics will help provide a larger market in which programmers who understand this technology can sell their skills. Since the largest consumer of graphics processors is the video board industry, I feel appropriate presenting an overview of the two leading graphic processors and how they are used in a video environment.

The two processors I am referring to are the Intel 82786 and the TI TMS 34010 (pronounced "three-forty-ten"). Both were announced almost three years ago, and both offer incredible functionality. The reason for the rising popularity of graphics processors is their ability to handle the substantial effort of drawing lines, circles, and other primitives, thereby freeing up the system processor for other tasks. This concept is known as *distributed processing*.

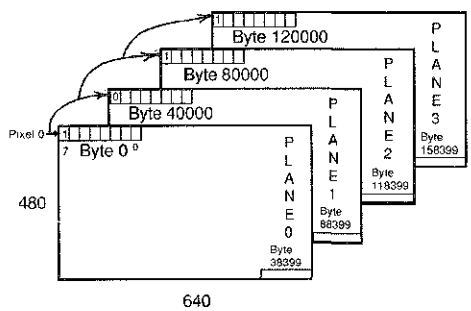
If you are familiar with the fundamental concepts behind EGA and VGA graphics, you know that the higher-resolution EGA/VGA modes use a "cross-plane" architecture. The cross-plane mode is a technique used by IBM to allow access to approximately 128K worth of graphics memory through a 64K window to reduce potential memory conflicts and properly utilize the segmentation of the 80x86 architecture. The other graphics modes, both the CGA-compatible and the special 8-bit mode, use a different architecture, known as *packed pixel*.

Plotting pixels in a cross-plane architecture generally requires that a couple of registers be set and, in some cases, that two memory writes occur. On the other hand, setting pixels in a packed pixel architecture may require a read/modify/write operation, but only when the pixel size is smaller than the data bus width.

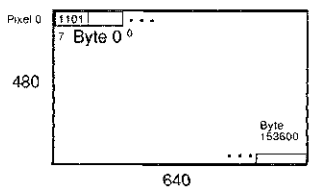
Most high-end graphics boards use the packed pixel architecture because it is more suitable for image-processing operations and because it tends to be easier to manage the much greater amount of memory required by high resolutions. This also applies to many graphics processors, including the 34010 and the 82786. In order to get around the 64K-segment problem, many of these high-end video boards use bank switching, which allows a block of memory, usually 16K or 64K, to be mapped into the PC's address space. If a program needs to address a pixel that is not accessible in the currently mapped block of memory, it just maps in the correct block and performs its work. This procedure is analogous to working with expanded memory. I was surprised that IBM didn't use this approach with the EGA when they were developing it, and now they have doomed themselves to maintaining backward compatibility.

One "standard" that seems to be different for each type of graphics product is pixel order in memory. Both the 34010 and the 82786 use an external data bus 16 bits wide; pixels are ordered within a word. The 82786 stores pixels starting in the most significant part of a word; just to confuse things, the 34010 stores

Pixel 0 is a 4-bit pixel with a value of 13_{10} (1101_2) in both cases (planar and packed).
Pixel can be selected by using a combination of bit and plane masks, and writes to plane 0.



Sample planar architecture



Sample packed pixel architecture

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

QEdit *Advanced*

the "quick editor"

By SEMWARE

QEdit is one of the smallest, fastest, most easy to use multi-file editors available for IBM Compatible Computers with many features to enhance your productivity.

- ***Extremely Fast***
- ***Completely Configurable (Including Keyboard!)***
- ***Easy to Use Macro Facility***
- ***Optional "Pop Down" Menu System***
- ***Recover Deleted Text***
- ***Edit Dozens of Files Simultaneously***
- ***Utilize up to 8 Windows to View Files***

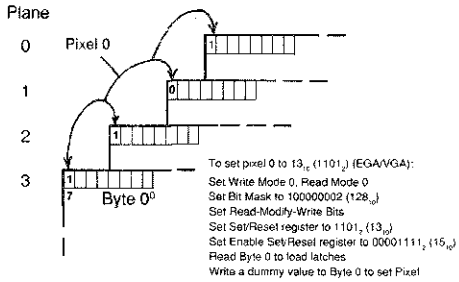
QEdit *Advanced* requires an IBM PC/XT/AT or Compatible, IBM PS/2 30-80, 128KB of available memory. Please specify 5.25" or 3.5" diskette.

$54.95 MASTERCARD & VISA ACCEPTED

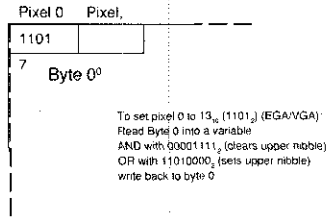
Add $3.00 Shipping & Handling. Georgia Residents Include 4% Sales Tax.

To Order:
SEMWARE, 730 Elk Cove Court
Kennesaw, Georgia 30144
Phone Orders: (404) 428-6416

Number 148 on the Reader Service Card


Plane
0
1
2
3
Pixel 0
Byte 0⁰
To set pixel 0 to 13₁₆ (1101₂) (EGA/VGA):
Set Write Mode 0, Read Mode 0
Set Bit Mask to 100000002 (128₁₀)
Set Read-Modify-Write Bits
Set Set/Reset register to 1101₂ (13₁₀)
Set Enable Set/Reset register to 00001111₂ (15₁₀)
Read Byte 0 to load latches
Write a dummy value to Byte 0 to set Pixel

Setting a pixel in a cross-plane mode (Y bits per pixel)


Pixel 0
Pixel,
1101
Byte 0⁰
To set pixel 0 to 13₁₆ (1101₂) (EGA/VGA):
Read Byte 0 into a variable
AND with 00001111₂ (clears upper nibble)
OR with 11010000₂ (sets upper nibble)
write back to byte 0

Setting a pixel in packed pixel mode

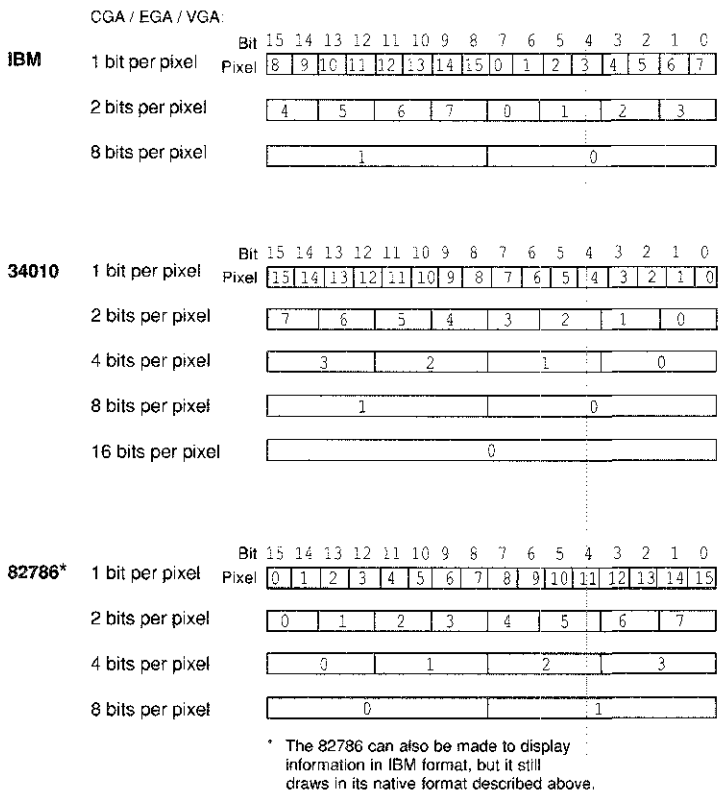
pixels starting in the least significant portion. [To make matters even worse, the PC standard (CGA/EGA/VGA) packed-pixel formats use another variation.] This makes a big difference when transferring data from one to the other.

This difference is just the start. Many people make the mistake of trying to compare the 82786 and 34010 head-on. This sort of comparison is invalid because the 34010 is a fully functional processor, whereas the 82786 is a coprocessor, similar in concept to a numeric coprocessor. This being the case, it's best to examine each processor in some detail.

The 34010

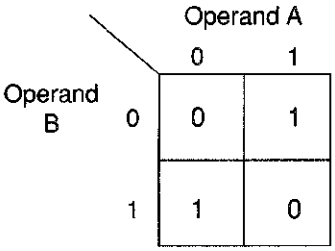
With VGA boards supporting graphics at 1, 2, 4, and 8 bits per pixel—and PC developers just starting to understand the possibilities—it may be surprising that the 34010 supports 16-bit-per-pixel drawing in addition to the depths supported by the VGA. The 34010's architecture lends itself to graphics because it addresses memory in bits. Conventional processors, such as the 8086, address memory in bytes. Bit addressing assists graphics operations by allowing individual pixels to be located and modified.

Of course, another component is required: data width. The 34010 provides a feature known as "fields." Each instruction that accesses memory must use one of the two field registers to determine how many bits wide the memory access should be. Fields can specify a width of 1 to 32 bits. Although this feature is transparent, it is not without some cost. Since the 34010's external data bus


CGA / EGA / VGA:
IBM
1 bit per pixel
2 bits per pixel
8 bits per pixel
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Pixel 8 9 10 11 12 13 14 15 0 1 2 3 4 5 6 7
4 5 6 7 0 1 2 3
1 0
34010
1 bit per pixel
2 bits per pixel
4 bits per pixel
8 bits per pixel
16 bits per pixel
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Pixel 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
7 6 5 4 3 2 1 0
3 2 1 0
1 0
0
82786*
1 bit per pixel
2 bits per pixel
4 bits per pixel
8 bits per pixel
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Pixel 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7
0 1 2 3
0 1
* The 82786 can also be made to display information in IBM format, but it still draws in its native format described above.

Visual packed pixel representations within a word (16 bits)

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



XOR Truth Table

$A = 37_{16}$. $B = F8_{16}$

A XOR B = CF_{16}

A XOR CF_{16} = $F8_{16}$ (same as B)

B XOR CF_{16} = 37_{16} (same as A)

Example of using XOR twice to recover previous values.

is 16 bits wide (the internal bus is 32 bits wide), any memory access that does not start on a word boundary and is not 16 bits wide will take slightly longer to execute.

The 34010 has two clock inputs: One, the pixel clock, drives the video output; the other is used as the system clock. TI is currently producing 40-, 50-, and 60-MHz 34010s. (Before you get all excited, note that the clock speed is divided by 8 before being used to time instruction cycles.)

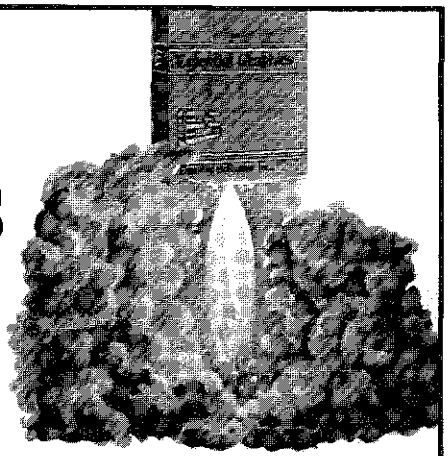
In addition to an instruction set somewhat reminiscent of the PDP-11's, the 34010 has several instructions that were designed specifically for graphics operations:

FILL	Fills a rectangular area of memory with the foreground color.
PIXBLT	Moves a rectangular block of memory to another memory location. PIXBLT stands for PIXel BLock Transfer.
LINE	Draws a line.
DRAV	Provides incremental drawing support. DRAV (draw advance) is used to implement graphics primitives, such as circles and arcs.

The 34010 has thirty-one 32-bit data registers, split into two groups. TI refers to these groups as "register files." Register file A contains the general-purpose registers, and register file B contains the function-dependent registers. One of these 31 registers, the stack

Power Graphics

Essential Graphics Takes You To New Heights Of Graphic Programming In C. Increases Speed 40%.



When first brainstorming this ad I spent a considerable amount of time trying to determine what graphic image to use as an illustration. The Space Shuttle, Mona Lisa, Robo-Cop - there are so many available.

Then it occurred to me. When you have the fastest, smallest functions, it's really irrelevant to show a complicated graphic image. It would be as if thinking up a sexy graphic were the test of a library.

The Graphics Test

The crucial test of a professional graphics package is: are the functions powerful, reliable, fast, and do they truly eliminate grunt work?

How quickly the functions execute is the criterion most people look for in a graphics library. There is no sense paying for a package that is not up to speed.

Beware Of Speed Traps

We eliminate the bios calls and write directly to the graphics card. As a matter of fact, in a recent benchmark, we were clocked 40% faster then our nearest competition.

I'd like to repeat that "...clocked 40% faster than our NEAREST competition." Please take a moment to think about the significance of that speed increase in the project you are contemplating or working on now.

Our efficient, granular coding provides you with code sizes up to 75% smaller. Lean, fast and tight - just the way you would have done it yourself.

Power Packed Pixels In Every Package

There has always been a trade-off in this industry between ease of use and power. Our functions do not require a lot of setups, are well-documented, and most of all, thoroughly debugged. Essential Graphics' ease of use stems from our thoughtfulness and not from a lack of power. We explain what we are doing every step of the way. Our support staff consists of the humans who wrote the functions, so we are thoroughly prepared to assist you after the purchase.

Essential Graphics is a trademark of Essential Software

Caveat Emptor

Make no mistake, this is not a package for the "draw a box around the total field" crowd. This library was designed to help the professional C programmer make money and look good.

We've included a complete set of "rubber-banding" functions. One of the most welcome features is the ability to save/restore images in PC Paintbrush format or bit image. World coordinates and view ports aid in programming portability.

We include the ability to manipulate and rotate character fonts and symbols. You can place characters and symbols anywhere on the screen, and use up to eight fonts at one time.

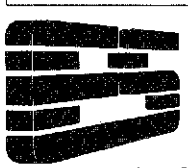
Yours, Mine, Ours

We don't consider ourselves equity partners in your business and therefore we do not charge any royalties or run time fees. We think your efforts belong to you. If for any reason you are unsatisfied with our product you may return it within 30 days for a full refund. Full source is available. Please call today and launch yourself into the world of power graphics.

Price $299 - Source $299

Adaptors include - CGA, EGA, VGA, MCGA, ATT, ATT DEB, Hercules, Vega Deluxe, Paradise Autoswitch. **Printer Support** - IBM, Epson, Oki, TI, Alps, Panosonic, and others. **Supports** mice, light pens, plotters, color printers. **Compilers**-Microsoft, Lattice and Turbo-C

Other Essential Products Include: ScreenStar - Essential Communications and Utilities -- /*resident_C*/ - Please call for further information 201-762-6965.



To Order Call:
1-800-451-6174
N.J. 201-762-6965
Fax 201-762-0118

Essential Software, Inc.
South Orange Plaza
76 S. Orange Ave., Suite 3
South Orange, N.J., 07079

Number 10 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

If you ever wanted to take a "crack" at assembly language...

Don't end up with egg on your face!!

THEY thought we had "cracked" when we said that our MS-DOS macro assembler clocked in with an assembly rate of 75,000 lines per minute on a 6MHz IBM AT. Yet we make no "yoke" that SPEED is just the beginning of OPTASM's features...

Now OPTASM has added ON-LINE HELP to make your job even easier.

OPTASM eliminates Phase Errors with its "n" pass design, and it automatically handles jumps out of range. Don't get "fried" with one or two pass assemblers that will give you NOPs, the dreaded Phase Error, and jumps out of range.

Have your assembly time "over easy", with the FAST, EFFICIENT, and PRACTICAL features that are packed into OPTASM!!

New ... OPTLNK, a linker that gives you spare time!

OPTLNK is *the fastest* linker for MS-DOS systems, up to ten times faster than MS-LINK. OPTLNK will handle LARGE files in a flash. Just like OPTASM, OPTLNK has full CodeView support! It provides superior error handling and a global cross-reference feature that you are going to love.

"If you are a professional assembly language programmer, this is a mandatory purchase"

-Barry Simon, PC Magazine.

So get cracking...

Free OPTASM Demo!!
Download from our
BBS: 412-282-2799

Call *today* for more information!
800-833-3061 (USA)

OPTASM: $125 list **OPTLNK:** $125 list
30 day money-back guarantee

SLR SYSTEMS
1622 N. Main Street Butler, PA 16001
412-282-0864 TELEX 559215

Number 165 on the Reader Service Card

pointer, appears in both register files, thereby creating the impression that each file has 16 registers. The graphics instructions do not use the field registers.

The B-file registers specify such things as the display start address, FILL dimensions, and foreground and background colors. The field registers referred to earlier are actually located in the status register, along with standard processor status flags. Another dedicated pair of registers in the B file consists of the source and destination pitch registers, which specify the number of bits that should be added to an address in order to end up at the same horizontal (x) position on the next sequential line. These are necessary for most graphical primitives that are not limited to operating on only one scan line. Unfortunately, a restriction on the pitch values requires that they be a power of 2; because of this, a 640x480 display will actually take up enough memory for a 1024x480 display, even though only 640 pixels will appear. This restriction is removed in the 34020.

Another set of registers, known as the I/O Control registers, are responsible for video control and some other special graphics operations, such as write-protecting bit planes and supporting logical writing modes. These registers are accessed by reading or writing to a specific area in the memory map.

Logical writing modes are arithmetic operations that act upon drawing functions. One commonly used logical writing mode is XOR mode. It is used to draw pixels in such a way that when the same sequence of pixel is drawn again, the contents of memory are restored to their original settings. This writing mode is used for drawing cursors, among other purposes.

Table 1 lists all the logical writing modes. The first 16 are also supported by the 82786.

Because the 34010 is really a 32-bit processor, it has the potential of addressing up to 4 gigabits (512 megabytes) of memory. (Actually, only 128Mb of memory can be addressed in the 34010, owing to some hardware limitations, but the 34020 will have access to the full 512Mb.) This may seem like an immense address space, but when you consider that a 640x480 8-bit display takes up 480K (remember the pitch limitation), you realize that graphics eats up memory altogether too quickly.

The 34010 has the advantage of being able to accept intelligent, nongraphics tasks from the PC. This means that in a properly designed application, you could put some of the device-independent functions (those that don't require access to PC devices, such as drives or keyboards) onto a 34010 while the PC goes off to do something else that may be more critical. I am not trying to imply that you can directly execute 8086 code on the 34010—code for the 34010 is best developed using the TI C Compiler and Assembler, a much more pleasant alternative to hand coding.

Although this is a rather rough overview of the 34010, it should give you a taste of its capabilities. My sources claim that 100 thousand 34010s were shipped last year (with that amount to double this year), and most of these chips are ending up on graphics boards. Some companies that utilize the 34010 on their graphics boards are TI, Number Nine, Renaissance GRX, Matrox, NEC, PC Tech, and Imagraph.

The Intel 82786

As I mentioned earlier, the 82786 is a graphics coprocessor, which, in this context, means that it is capable of executing hard-wired graphics primitives but not general-purpose code. The disadvantage of a coprocessor is obvious: You need an external processor to feed the 82786 its commands. The external processor also has to run the code that performs the more complex graphics operations, such as flood fills. This will obviously consume more of the host processor's bandwidth, but it still outperforms working with a "dumb buffer" (unassisted) board.

The 82786 is, surprisingly enough, a more complex design than the 34010. This is due to the '786 actually consisting of three separate (but interacting) processors: the Graphics Processor (GP), the Display Processor (DP), and the Bus Interface Unit (BIU). The BIU controls all access to the chip as well as to the memory it manages (up to 4Mb). The accesses can come from three sources—the GP, DP, and any external processors—and they can be assigned priorities, as well.

The GP is the processor that actually handles all the drawing primitives. The GP's hard-wired primitives are fast, because the chip was designed specifically with that set of functions in mind. In addition to the listed primitives, the '786 supports line patterns, the 16 logical writing modes discussed earlier, a plane mask, and pixel transparency. These operations are all supported at 1, 2, 4, and 8 bits per pixel.

continued . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

TurboC, Lattice C, and QuickC

"c_wndw" Windows, Menus & File Access

Now there is an affordable, complete set of functions to produce end-user DOS applications in the C language. "c_wndw" is a very complete toolkit, providing the following facilities

- o Color and cursor control
- o Pull-down and pop-up menus, and dialog lines
- o Field formatting and string handling
- o Entry editing of numbers and dates
- o Stacked, overlapping windows
- o File access error trapping and handling
- o Access to dBASE III plus '.DBF' files
- o Hashed random file access
- o Function key mapping and control
- o Undo and Help capability

For a license fee of $35, we provide a full set of object libraries for a C compiler, the full manual, and 19 sample programs. (S&H $2, GA res. add $1.85 tax). Specify your C compiler. Source code is available for $95.

To find out more about the "c_wndw" toolkit, write or call for information to Marietta Systems - (404) 565-1560
P.O. Box 71506, Marietta, GA 30007 Visa-M/C

Trademarks: "c_wndw", dBASE, QuickC, Turbo C, Lattice C

Number **223** on the Reader Service Card

34010 Graphics Co-Processor for the PC/XT/AT Bus

The TI 34010 is the best news to come along in the graphics processor arena in a number of years. PC Tech, Inc. has developed several TMS 34010 based PC compatible display adapters that allow the full power of the TMS 34010 to be used on PC/XT/AT compatible machines.

MONOCHROME

- 256k display memory (384k optional)
- 512k program memory (2M optional)
- Up to 48 mHz 6 MIPS operation
- Both ECL and TTL monitor drivers
- Multiple programmable dot clock frequencies up to 150 MHz
- Maximum resolution 2048 x 1536
- Capable of drawing 125 million pixels per second
- CGA / MDA / Hercules Emulation
- Suggested list price $795 ($1350 with full page black on white monitor.)

COLOR

- Up to 1 Mbyte display memory (8 bit / pixel)
- 1 - 4 Mbyte program memory
- 256 simultaneous colors from palette of 256,000
- Up to 48 mHz 6 MIPS operation
- Dual (piggyback) single slot board
- Multiple programmable dot clock frequencies
- Graphics resolution 1024 x 768 x 256
- Custom dot clocks available
- Suggested list price $1350 (512K video memory, 1 Mbyte program memory)

TI 34010 Assembler, Debugger and "C" Compiler available!

Both boards support **DGIS** from Graphic Software Systems, Inc. Most major text and graphics applications are supported on both boards via **DGIS** and low level graphics interfaces. Custom implementations of either design and private labeling are also available.

Contact:

PC Tech

907 N. 6th Street
Lake City, MN 55041
612-345-4555
FAX: 612-345-5514

PC, XT and AT are trademarks of International Business Machines Corp.
DGIS is a trademark of Graphic Software Systems, Inc.
Prices subject to change.

Number **208** on the Reader Service Card

C Programming Tools from ITHACA STREET SOFTWARE, INC.

FONT-TOOLS

If you work with GEM screen fonts, or use a graphics package with limited font support, then you should consider getting FONT-TOOLS.

* *Powerful mouse-driven font editor with dozens of built-in functions and effects - create italic, bold, shadowed, and outlined typefaces.*
* *Utilities for resizing font files, downloading GEM screen fonts to a LaserJet, viewing fonts, more.*
* *Dozens of sample font files included - bundle with your application or use as models for creating new fonts.*
* *C function library for displaying fonts from a graphics application - use with your compiler's built-in graphics library.*

Single User Price: $99.95
Developer's Edition: $149.95

Baby Driver

NOW YOU CAN ADD FULL-FEATURED PRINTER SUPPORT TO YOUR GRAPHICS APPLICATION!

* *Supports over 150 dot matrix, inkjet, and laser printers. Add additional support yourself with our full-screen PDT editor.*
* *Print specified areas of the screen or print from in-memory images. Portrait and landscape orientation. Image scaling, color rendition via shading patterns, more.*
* *Linkable object libraries containing almost 40 C callable functions - this is not a TSR program. Adds only 12K to your program.*
* *All printers selectable at run time, or link in only designated devices.*

Single User Price: $99.95
Developer's Edition: $199.95

ICON-TOOLS/Plus

An enhanced edition of our popular ICON-TOOLS product!

* *New, improved version of our icon editor and modify utilities.*
* *Additional icon files and sample programs. Icons now include several fonts in different styles.*
* *MOUSE SUPPORT for Quick C/5.0 and Turbo C. Now you can build mouse-selectable icon menus.*
* *Enhanced function library. Still includes full source.*
* *Non "Plus" version still available for only $79.95.*

Single User Price: $99.95
Developer's Edition: $125.00

SALES: 13103 Travis View Loop * Austin, TX 78732 * (512) 343-5064
SUPPORT: 1145 Ithaca Drive * Boulder, CO 80303 * (303) 494-8865

Number **242** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

If you ever wanted to take a "crack" at assembly language...

Don't end up with egg on your face!!

THEY thought we had "cracked" when we said that our MS-DOS macro assembler clocked in with an assembly rate of 75,000 lines per minute on a 6MHz IBM AT. Yet we make no "yoke" that SPEED is just the beginning of OPTASM's features...

Now OPTASM has added **ON-LINE HELP** to make your job even easier.

OPTASM eliminates Phase Errors with its "n" pass design, and it automatically handles jumps out of range. Don't get "fried" with one or two pass assemblers that will give you NOPs, the dreaded Phase Error, and jumps out of range.

Have your assembly time "over easy", with the FAST, EFFICIENT, and PRACTICAL features that are packed into OPTASM!!

New ... OPTLNK, a linker that gives you spare time!

OPTLNK is *the fastest* linker for MS-DOS systems, up to ten times faster than MS-LINK. OPTLNK will handle LARGE files in a flash. Just like OPTASM, OPTLNK has full CodeView support! It provides superior error handling and a global cross-reference feature that you are going to love.

"If you are a professional assembly language programmer, this is a mandatory purchase"

-Barry Simon, PC Magazine.

So get cracking...

Free OPTASM Demo!!
Download from our
BBS: 412-282-2799

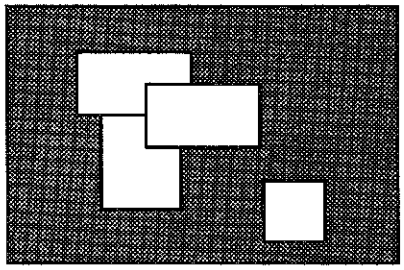
Call *today* for more information!
800-833-3061 (USA)

OPTASM: $125 list **OPTLNK:** $125 list

30 day money-back guarantee

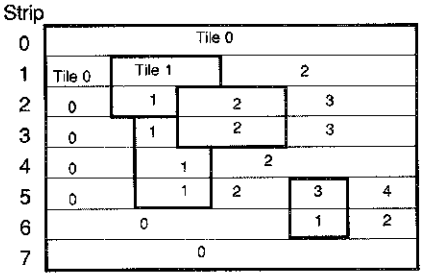
SLR SYSTEMS
1622 N. Main Street Butler, PA 16001
412-282-0864 TELEX 559215

Number **165** on the Reader Service Card



Shaded area is field, i.e., not data.

What you see on screen, 4 hardware windows



8 strips, with 1 to 5 tiles each

Strip and tile breakdown of an 82786 screen

programmability is severely limited. It does have CALL and RETURN commands, though; so you could set up a generic command list to, for example, draw a box with an *X* in it at the current *x,y* position and make it a subroutine that you could call in from your 82786 "program."

The Display Processor is probably the most underutilized feature of the chip. Most graphics-board companies using the 82786 use the DP for driving the monitor output and not much else. But the DP provides two other features: a hardware cursor and hardware windowing.

Hardware windows are similar in concept to the overlapping software windows utilized in such packages as Microsoft Windows. The big difference is that hardware windows do not require a redraw of screen information when the window is moved. This means that you can have virtually simultaneous drawing going on in a number of overlapping windows without worrying about clipping, because the hardware windows only display what's in memory; they do not alter memory. Hardware windows can also display *field*, a term used to indicate non–memory consuming display information ("dead space"). Because all normal limitations are removed, the long-time tie between memory and display is completely altered, requiring programmers to toss out everything they know about displaying PC-based graphics.

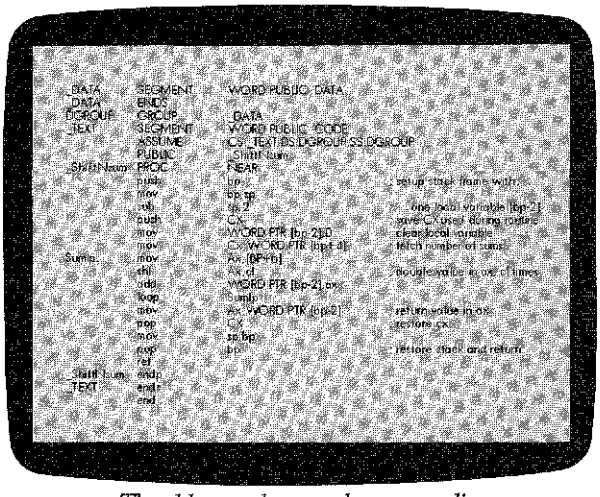
Of course, you can still use the one-to-one correlation between memory and display by creating a full-screen hardware window. Hardware windows are rectangular tiles that display data from memory at 1, 2, 4, or 8 bits per pixel. To top that off, a window can be flagged to indicate to the hardware that it should be zoomed. The zoom factor ranges from 1 to 64 in both the *x* and *y* directions. The screen position of a hardware window as well as the contents are determined by the DP display list. This display list consists of a series of strips, each of which contains a number of tiles. There is a limit of 15 tiles per strip, but no limit on the number of strips, providing each one is at least one scan line high.

Although the 82786 is a coprocessor, it does have three sets of registers that can be accessed by an external processor. One set belongs to the DP and is used for setting display frequencies, display list location, cursor pattern, and so forth. Another set belongs to the GP and contains information regarding the current drawing environment, such as *x,y* position, colors, and start of drawing area in memory. The last set of registers is the set containing the processor control. This set can be either memory

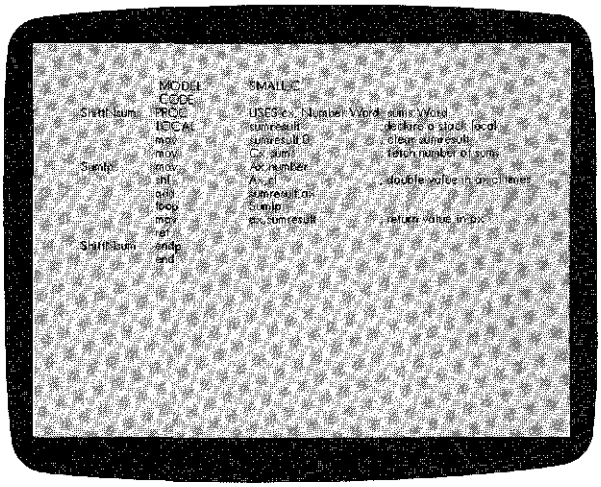
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Our new MASM 5.1 gives you two things you've never had in a macro assembler.

MS OS/2 and spare time.



The old way: slow, cumbersome coding.



The new way: time off for good behavior.

As a professional programmer you know how fast MASM subroutines make your programs run. But you've also had to pay the price of cumbersome coding.

With Microsoft® Macro Assembler version 5.1, you get ease *and* speed. Starting with our *Mixed Language Programming Guide* that teaches you how to call MASM subroutines from your high-level programs. To the PROC directive that makes it as easy to pass parameters to an assembler subroutine as it is in high-level languages. Plus you get our easy-to-use "fill-in-the-blanks" template programs.

In addition to MS-DOS® support, MASM 5.1 has full MS® OS/2 support that lets you do things like break the 640K barrier, create multithreaded applications and Dynamic Link Libraries. And you get a complete set of high-octane development tools that lets you work faster under both environments. As well as the reconfigurable Microsoft Editor, a high-speed linker and new incremental linker for MS OS/2. And the renowned Microsoft CodeView® window-oriented source-level debugger that lets you blaze through debugging.

For the details, check the techbox. For more information, call (800) 541-1261, Dept. I17. And learn how today's new Microsoft MASM is so fast and easy it comes with the following offer.

The night off.

Microsoft Macro Assembler Version 5.1 for MS OS/2 and MS-DOS

Mixed-Language Programming Support and Documentation
- Mixed-Language Programming Guide with step-by-step examples.
- MASM templates for interfacing to BASIC, C, FORTRAN, and Pascal.
- Sample macros for interfacing for DOS and OS/2 systems. NEW!
- Simplified segment directives make writing MASM subroutines easy.
- Comprehensive instruction set *reference* with examples.
- Programmer's Guide – complete guide to MASM features and instruction set.

Macro Assembler for MS OS/2 and MS-DOS
- Full MS OS/2 support to break the 640K barrier. NEW!
- Supports the 80386 and 80387 instruction set.
- Assembles 25-40% faster than MASM 4.0.
- Have labels local to a procedure. NEW!
- Declare stack based local variables within a procedure. NEW!
- OS/2 systems incremental linker – up to 20 times faster than a full link. NEW!

Microsoft CodeView Window-Oriented Debugger and Microsoft Editor
- Full OS/2 systems support. NEW!
- Debug MASM code using original source code with your comments.
- Interactively view structures and chase linked lists. NEW!
- Take advantage of the 80386.
- Set dynamic breakpoints, watch variables, memory, registers and flags.
- MS OS/2 and MS-DOS reconfigurable programmer's editor. NEW!

Microsoft®

© 1988 Microsoft Corporation. Microsoft, the Microsoft logo, MS, MS-DOS and CodeView are registered trademarks of Microsoft Corporation.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

mapped or I/O mapped and allows the external processor to initialize the chip and send commands to the GP and DP.

Some companies that offer boards based on the 82786 are Autograf, Bell Technologies, and Verticom. Also, Number Nine offers a board that sports both an 82786 and a 34010.

Interfacing with the Processors

As you may be able to tell, there is a big difference in the way these processors are programmed. In order to take full advantage of their special features, you must access them directly.

If you do not need these special capabilities but still want to access the high drawing speed that these processors offer, you may want to consider some interfaces offered by various companies. These interfaces include such packages as MetaWindows by Metagraphics, DGIS and CGI from Graphics Software Systems, and HALO by Media Cybernetics. The problem with using packages like these is that they have to support a large variety of graphics products, all with different capabilities; therefore, they have had to use the most common features to define their interfaces. On the other hand, board-specific software interfaces, such as Renaissance GRX's RGDI or Number Nine's NNIOS, give you access to board-specific features, but only for the boards of their respective companies. One newly announced interface, TIGA-340 from TI, may help standardize the writing of software for boards based on the TI 340X0 series of processors.

Looking Ahead

We know where TI is going with their graphics-processor line: The 34020 has a 32-bit bus (internal and external) and supports faster drawing operations and a floating-point processor, the 34082—you can actually attach up to seven of them. It has a slew of other enhancements, as well. The question is, Where is Intel heading? At this point, Intel has given no official word, but I have a feeling we'll be seeing something high powered and graphics related in the near future.

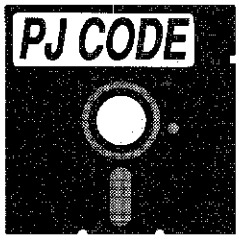
Numerous companies have realized that the graphics-processor market is maturing into a full-fledged industry and have, therefore, come up with their own entries into the market. Among these are several RISC processors from such companies as AMD and Weitek, as well as some drawing accelerators from National Semiconductor and Cirrus Logic. Information has appeared in trade journals about silicon-level clones of the 8514/A. None of these has been taken too seriously by board manufacturers yet, but only time will tell for sure.

EGA and VGA may be the current graphics hardware standards, but they are limited in their growth. New hardware with high-speed drawing abilities is required to allow users to migrate to higher resolutions and greater color selection. The hardware is already on the horizon, but it needs support in the form of software that takes advantage of the capabilities offered. ♦

Jake Richter is president of Panacea, Inc., a PC consulting firm specializing in graphics processors. Readers may contact Jake on MCI Mail or BIX, user name JRICHTER.

Texas Instruments, Inc.
Mail Station 712
P.O. Box 1444
Houston, TX 77251
713-274-2340

Intel Corp.
P.O. Box 58126
Santa Clara, CA 95052-8126
800-468-3548



Git yer *PJ Code DiskScription!*

HOT CODE—
COOL PRICES—WE DELIVER!

Typing in long code listings uses up a lot of valuable time, and calling up bulletin boards doesn't appeal to everyone. So what does the modern programmer do? Get a ***PJ Code DiskScription***, of course!

What's a PJ Code DiskScription? All the code listings from six consecutive issues of *Programmer's Journal*, sent directly to you every other month on disk, for the money-saving subscription price of only $39.95.*

You'll get all the upcoming installments of Michael Abrash's EGA/VGA ASM code, more of Tom Swan's handy Pascal code, plus additional great code from more of the country's hottest programmers.

You'll receive your code disks in the mail every other month throughout the year.

Learn from ***PJ*** code or use in it your own programs. Each disk contains all of the program listings published in ***PJ*** plus .EXE files, .OBJ files, benchmark source files, and more!

If purchased separately, the regular price for these 6 disks is $60.00. **So save yourself time, effort and over $20.00—order your *PJ DiskScription* today!**

***Programmer's Journal PJ Code DiskScription*—**
Code disks from 6 issues for only
$39.95*

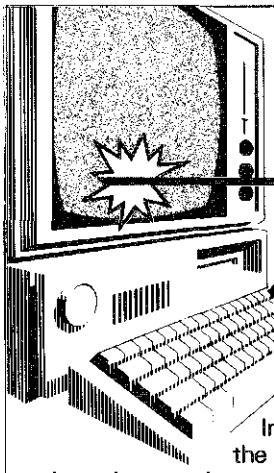
* Prepaid. US only. Call or write for prices outside the US.

*GUARANTEE: If you are dissatisfied with your **PJ Code DiskScription** for any reason, we will gladly send you a refund on all unmailed diskettes.*

GET HOT *PJ* CODE DELIVERED DIRECT TO YOU.
START YOUR *PJ Code DiskScription* TODAY!

PJ Code DiskScriptions must be prepaid.
For faster service call toll-free: 800-234-0386 with your VISA or MASTER CARD.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Get To Know Your Programs Inside! and Out!

Now you can analyze your programs with unprecedented detail with Inside!, a new software package from Paradigm Systems.

Inside! allows you to examine the route your programs take through execution counts, minimum, maximum and total elapsed times and a count of how many times each source line executes—function by function—for popular Borland and Microsoft languages!

Now available

Inside! Turbo C
Inside! Turbo Pascal
Inside! WATCOM C
Inside! Quick C
Inside! Quick Basic
Inside! Microsoft Pascal
Inside! Microsoft Fortran
Inside! Lattice C
Inside! Logitech Modula-2
Inside! TopSpeed Modula-2

PARADIGM SYSTEMS INCORPORATED

Paradigm Systems Incorporated $75.00 Each

P.O. Box 152 Milford, MA 01757

To Place Orders (800)537-5043 Product Support (508)478-0499

Visa/Mastercard Accepted

Inside! is a trademark of Paradigm Systems Incorporated.

Number 195 on the Reader Service Card

Make it with the best!

Speed program development time with OPUS Make.

OPUS Make tracks your executables and libraries better than any other Make utility on the market:

- DOS and OS/2 executables.
- DOS version uses **only 3K of memory!**
- Read and Run-time conditional directives.
- Loop, Include and Error directives.
- Multiple directory support.
- Extensive macro capabilities.
- DOS version supports OS/2-like standard error redirection.
- POLYTRON PVCS logfile support. Burton TLIB and MKS RCS support coming.
- UNIX makefile syntax.
- Microsoft makefile compatibility mode.
- **Also comes with OPUS MKMF:** An automatic dependency generator that fully understands C preprocessor directives.

OPUS Make and MKMF are writtten by the author of NDmake.

$79 + $3 S&H

OPUS Software

1032 Irving St. #439, San Francisco, CA 94122

(415) 664-7901

Number 254 on the Reader Service Card

NEW VERSION 2

MICROSOFT, TURBO AND MIX POWER C PROGRAMMERS... C WINDOWS TOOLKIT™ PUTS YOU IN CHARGE OF VIDEO *and* DATA-ENTRY

NEW FEATURES

- *Comprehensive data-entry and verification procedures with scanf( ) type syntax.*
- *Predefined data-entry functions for dates, SSNs, telephone numbers, currency amounts with commas, time, scientific notation, etc.*
- *Turn any window into an editor with your choice of keyboard mapping (Wordstar™ by default).*
- *Virtual windows up to available memory.*
- *Save windows to disk.*
- *Nested windows.*
- *Create and edit VGA/MCGA/EGA/Hercules fonts of any size.*

WINDOWING FUNCTIONS

- Create pop-up windows
- Create pull-down menus
- Create spreadsheet menus
- Create context-sensitive help screens
- Store windows in RAM or on disk
- Modify windows out of view
- Use 8 different types of exploding windows
- Scroll and move windows

MicroWay 386 SUPPORT*

- Runs in protected mode
- Uses 32-bit instruction set

*Extra cost option-call

SYSTEM CONTROL

- Device-independent output
- Supports VGA/MCGA/EGA/Hercules/CGA/MDA
- Detect the types of video adapter installed
- Switch between adapters
- Detect ANSI.SYS
- Control the size and position of the cursor
- Detect the enhanced keyboard
- Disable the video signal
- Delay program execution to microsecond resolution

VGA/EGA SUPPORT

- Use all 64 VGA/EGA colors
- Use VGA 29/43/50 line modes
- Use EGA 43-line mode
- Use 2 fonts simultaneously
- Design and edit custom fonts with FONTEDIT,™ our font editor (included)
- Smooth scroll the screen
- Smooth pan the screen

FAST SCREEN I/O

- Write formatted output (like **printf( )**)
- Get snow-free output on the CGA
- Scroll the screen
- Use direct screen writes or the BIOS

HERCULES SUPPORT

- Create and edit your own Ramfonts™
- Detect the presence of a Hercules Card
- Detect Ramfont™ support
- Load and store Ramfonts™ to RAM or disk
- Switch between modes
- Detect whether the Hercules Card is in text or graphics mode

Full source code included — No run-time royalties.
Includes 230 page manual — 30-day Money-Back Guarantee

Magna Carta SOFTWARE

P.O. Box 475594
Garland, TX 75047-5594
FAX: (214) 226-0386

Requires: IBM PC, XT,AT, PS/2 or compatible.
Supports Microsoft C 5.0/5.1/Quick C, Borland Turbo C 1.5, Mix Power C

CALL (214) 226-6909

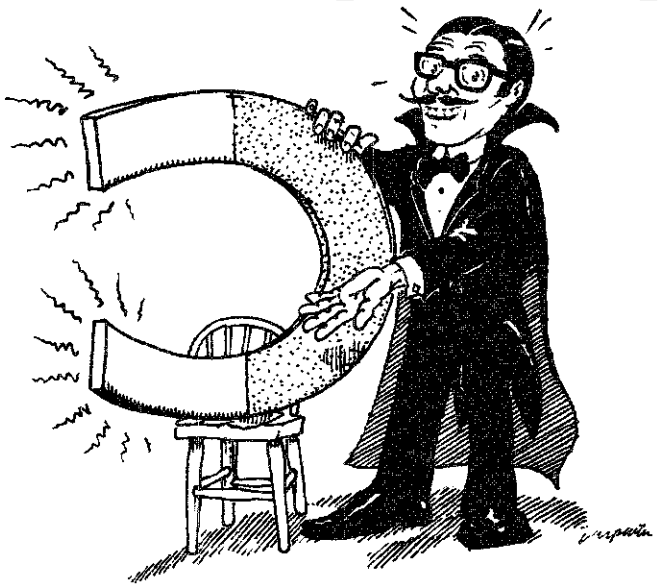
MasterCard VISA **Only $99.95** (Texas Residents add 8% sales tax.)

Number 212 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Demagnetizing Static Variables

by
Tom Swan



Modifying typed constants (alias static variables) stored in Turbo Pascal 4.0 and 5.0 compiled EXE code files is hairy business—but it can be done if you're careful.

Static variables, or typed constants as Turbo Pascal calls them, stick to compiled code like magnets to iron. Declared as constants but compiled as variables, typed constants are stored on disk inside a program's EXE code file. Because typed constants have predefined values, they eliminate wasteful initializations of global variables at runtime. Despite this advantage, however, after compiling, there's no easy way to change a program's typed constants—for example, to let people make keyboard assignments, select window colors, and modify other default settings.

A telephone call from a reader prompted me to think of several possible solutions to this problem. I knew I'd have to locate the typed constants embedded in the compiled code file and modify only the necessary bytes, storing the result back on disk with all other bytes intact. Mucking around with EXE disk files gives me the heebie-jeebies, so the code would have to be reasonably immune to changes in future compiler versions. The program should run fast, be easy to modify, work with both small and large applications, wash the floor, dust the shelves, and take out the garbage.

The following describes the results of my experiments, largely successful although I'm still doing the floor, shelves, and garbage by hand. After a brief refresher course on typed constants as stored on disk and in memory, I'll explain how to use the accompanying library unit to write custom installation programs. Also included are demonstration programs that you can use as starting places for your own projects.

A Needle in a Byte Stack

Finding typed constants in memory and on disk isn't nearly so difficult as finding a needle in you-know-what. Let's start with values in memory, as these are the easiest to locate. The first typed constant in memory is always stored at offset zero in the data segment. (There is only one data segment in a Turbo Pascal program.) Suppose you have these declarations:

```
const
  aword : word = 1024;
  achar : char = 'X';
  abyte : byte = 15;
```

The typed constant **aword** is stored at DSeg:0000, occupying two bytes. After this comes **achar** at DSeg:0002. Next, **abyte** follows at DSeg:0003. **DSeg** is a predeclared constant equal to the value of register DS. Rather than calculate these offsets manually, use the **Ofs** function to find the address of any typed constant. For example, this displays the offset addresses of the three typed constants in memory:

```
writeln(
  ' aword=', ofs(aword),
  ' achar=', ofs(achar),
  ' abyte=', ofs(abyte) );
```

In Turbo Pascal 4.0, typed constant values follow each other with no wasted bytes in-between. As a result, lone byte values can lead to extra CPU machine cycles by forcing words to begin at odd addresses. If you insert a fourth typed constant word between **achar** and **abyte**, the new value resides at DSeg:0003, making the CPU work harder to access the value. To fix this problem, Turbo Pascal 5.0 provides a new switch, **{$A+}**, to align typed constants and other variables on even addresses, letting the CPU access 16-bit quantities as quickly as possible. (Word alignment offers no advantage on PCs with 8088 processors.) The default is **{$A-}**, for no alignment.

We Are Not Alone

Your typed constants are not alone in memory. The System unit, containing Turbo Pascal's runtime library and linked into every compiled program, declares its own set of typed constants. (The 5.0 *Reference Guide* lists these on page 125.) In memory, items in the data segment follow this order:

1. Program typed constants
2. Unit typed constants
3. System typed constants
4. Global variables

The main program's typed constants are always first, beginning at offset zero. Next come any typed constants, declared in units that the program lists in a USES declaration. Then come the System unit's typed constants, followed by the program's global variables. Together, these four items make

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

up the program's data segment, which can be as large as 64K. Items 1–3 are stored in the compiled EXE code file. Memory for global variables (item 4) is allocated at runtime—global variables are never stored in the disk code file.

Smart Linker; Dumb Programmer

Turbo Pascal's smart and crafty linker, which attempts to optimize compiled code, requires you to refer to at least one typed constant declared in your program and in all units. If you don't, the linker realizes you aren't using the typed constants and does not reserve space for them in the code! While writing codeless test programs to determine where typed constants are stored, it took me longer than I care to reveal to realize the linker was "helping" me by throwing my constants away. The following short program demonstrates how Turbo's smart-aleck linker can get you into trouble, as it did me:

```
program test;
const
  atc:char='a';
begin
  writeln( atc );
  writeln(
   char( ptr(dseg,0 )^) )
end.
```

The program declares one typed constant, **atc**. Two **writeln** statements display **atc**'s value, the letter *a*. The first **writeln** refers to **atc** by name. The second refers to **atc** the hard way, dereferencing a pointer to DSeg:0000 and recasting as type **Char**, admittedly the long way around a simple job. Take out the first **writeln** statement and guess what happens. The second **writeln** no longer displays *a*! Turbo Pascal throws out **atc** because it doesn't realize that the **ptr** expression refers to the typed constant.

Obviously, you won't usually refer to typed constants this way, but you might write a program that expects another module to reference a typed-constant array or large record at DSeg:0000. Don't do this. Turbo Pascal does not save typed constants unless you refer by name to at least one in each module. The same rule goes for typed constants in units. Either the unit code or another module must refer to at least one typed constant by name, or else the linker throws these babies out with the bathwater.

Sniffing for Typed Constants

Turbo Pascal stores typed constants as the last bytes in a disk code file. At runtime, the bytes are copied to the start of the program's data segment in memory. On disk and in memory, the bytes are paragraph aligned—that is, beginning at an offset in the code file and at an address in memory evenly divisible by 16.

The problem of writing an installation program to modify default values in compiled disk code files requires finding the offset to the first byte of the first typed constant. Listing 1 (WINDGLOB.PAS) makes this easy by declaring a marker *CBase* assigned the string value '@CBASE@'. The program's default settings follow **CBase**—in this sample, the typed constants **WBForeColor** to **WTitle**. Last, a dummy typed constant **EBase** marks the end of the typed constants area. **Ebase**'s data type and value are unimportant, but **CBase** must be a string.

Because the application and installation program refer to the same values, it's best to declare all typed constants in a separate unit similar to WINDGLOB.PAS. In this example, the unit has no code, although it certainly could.

The next step is to write the main application, WINDTEST.PAS in Listing 2. This program lists WINDGLOB in a USES declaration, importing the typed constants from Listing 1. Run WINDTEST to display a simple window with scrolling random characters. Press any key to end the program.

The last job is to write the installation program, which modifies the typed constant default values in the compiled WINDTEST.EXE program. Listing 3 (WINDINST.PAS) displays a menu to let you change window colors and type a new window title string. The next section describes how to compile and run the programs.

Compiling the Test Programs

First compile Listing 4, TCUNIT.PAS, creating TCUNIT.TPU. (With the integrated Turbo Pascal compiler, be sure the Compile menu's Destination option is set to "Disk.") TCUNIT has procedures to help you write your own installation programs and contains a function to locate and modify typed constant values in compiled EXE files. I'll explain how to use the unit in a moment.

After compiling TCUNIT, compile WINDGLOB.PAS to WINDGLOB.TPU. Then compile WINDTEST.PAS and run the test application. Finally, compile WINDINST.PAS and run it. Change any of the values listed in the menu. In this no-frills version, you have to specify colors by number even though WINDGLOB.PAS declares the default colors by name, using identifiers declared in the standard **Crt** unit. Press Q to quit and modify WINDTEST.EXE. Press X to quit and not save changes. Run WINDTEST again to see the effect of your new values.

How TCUnit Works

Two procedures in TCUNIT, **GetWord** and **GetStr**, prompt for word and string values. WINDINST calls these procedures to let you enter new values for various typed constants. You'll probably want to add your own procedures to prompt for values of other types, such as **GetInteger**, **GetChar**, **GetReal**, and so forth. Use the two procedures listed here as guides.

Function **ChangesSaved** in TCUNIT does all the work of opening the compiled EXE code file, locating the typed constants, and writing the modified values back to disk. See the end of WINDINST for an example call to this function, which requires a file name, the **CBase** marker string, and the offset addresses of the **CBase** and **EBase** typed constants. **ChangesSaved** performs several operations:

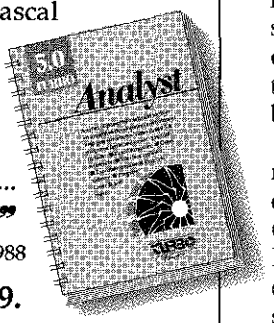
First the EXE file is opened. Parameter **fileName** must be the name of a compiled EXE file with **CBase** and **EBase** typed

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Write Better Turbo 4.0 5.0 Programs...

Or Your Money Back

You'll write better Turbo Pascal 5.0 programs easier and faster using the powerful analytical tools of **Turbo Analyst 5.0**. You get an integrated development environment with • Pascal Formatter • Cross Referencer • Program Indexer • Program Lister • Execution Profiler • Includes complete source code. ***+ a new execution timer!***



"Turbo Analyst...a valuable tool for every Turbo Pascal programmer. The Program Structure Analyzer... a timesaver. The integrated environment...impressive."

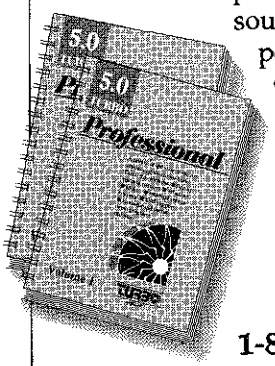
Namir Shammas, Turbo Tech Report, July/Aug. 1988

Turbo Analyst 5.0 is only $99.

Now with data entry screens!

A Library of Essential Routines

Turbo Professional 5.0 is a library of more than 500 state-of-the-art routines optimized for Turbo Pascal 5.0. It includes complete source code, comprehensive documentation, and powerful and useful demo programs. You get • TSR management • Menu, window, and data entry routines • BCD • Large arrays and more.



"Turbo Professional...a superbly crafted toolbox... [makes] the hard stuff so easy."

Kent Porter, Dr. Dobb's Journal, April 1988

Turbo Professional 5.0 is only $125.

Call toll free for credit card orders.

1-800-538-8157 ext. 830 (1-800-672-3470 ext 830 in CA)

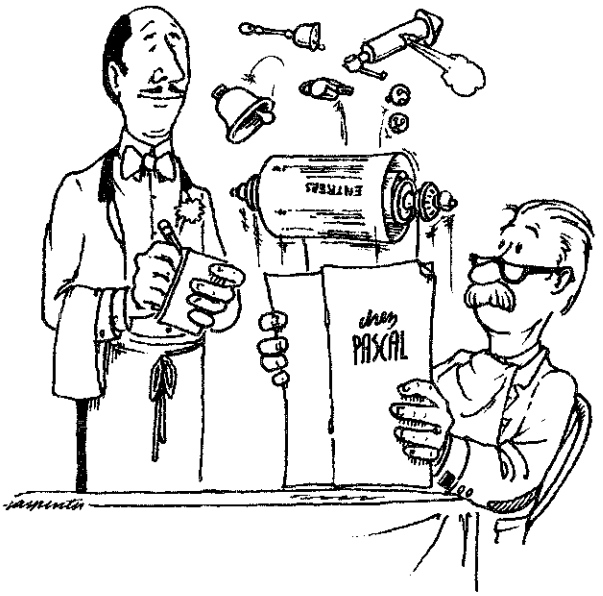
For upgrade or other information, call 408-438-8608.

Satisfaction guaranteed or your money back within 30 days. Turbo Pascal 4.0 or 5.0 is required. Shipping & taxes prepaid in U.S. & Canada. Elsewhere add $15 per unit.

TURBO Power

TurboPower Software
P.O. Box 66747
Scotts Valley, CA 95066-0747

Number **25** on the Reader Service Card



constants around the values to be modified, as shown in the listing of WINDGLOB.PAS.

Next, function **FoundCBase** returns **true** if the **CBase** marker string is found in the EXE code file. If so, **FoundCBase** returns the byte offset to this string. If not, **FoundCBase** returns **false**, causing an error message to be displayed.

If **CBase** is found, procedure **SaveData** copies the in-memory typed constants to the compiled EXE code file, overwriting only the bytes from **CBase** to the byte just before **EBase**. No other bytes are changed in the disk code file.

For all this to work correctly, you must be sure to use the identical typed constants in both the main (WINDTEST) and installation (WINDINST) programs. The safest bet is to declare all typed constants in one separate unit (WINDGLOB) and then use the unit in both programs.

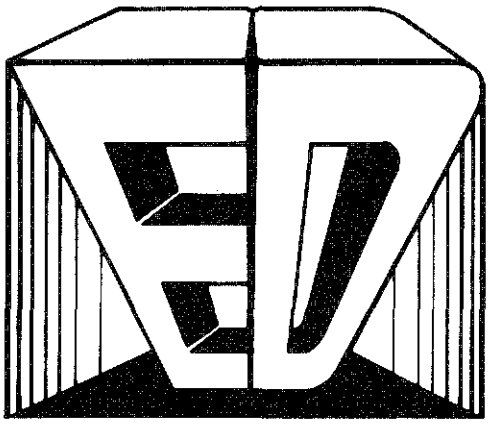
Writing Your Own Installers

Even without fully understanding every detail about how TCUNIT operates, it's easy to write your own installation programs. Just follow these steps:

1. You don't have to modify TCUNIT, although you might want to add procedures to prompt for specific data types, as mentioned earlier.
2. Store all typed constants in a separate unit and compile. Start with **CBase** and end with **EBase**, as in the listing WINDGLOB.PAS. The number, type, names, and values of typed constants in between **CBase** and **EBase** are completely up to you.
3. Write your application, using the unit of typed constants from step 2. Compile to disk.
4. Write your installation program, using WINDINST.PAS as a guide. Add TCUNIT and your typed constants unit to the program's USES statement. Call **ChangesSaved** to write modified values to your program's code file.

By the way, you can also use WINDINST to modify its own code file, WINDINST.EXE. This displays the modified settings the next time you run the installation program. Also, the **CBase** marker string (see WINDGLOB.PAS) can be anything you like—it doesn't have to be '@CBASE@', as it is here. The string must be unique, occurring nowhere else

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



EDITING SYSTEM™

ED is an editor that makes the very conception of other editors obsolete.

ED is an object-oriented open architecture system for editing programs and manuscripts, manipulating data files and creating snazzy demos. ED's commands respond with no perceptable delays. Programmers have complete simultaneous control over multiple source files. Functions for creating manuscripts are part of ED's design, with stream blocks, word-wrap, tabs, intelligent paragraph formatting, block justification, and pagination that behave correctly and execute instantly. ED can be used to create sophisticated sorts and filters. Macros, search and replace, column blocks, and block sorts allow records to be sorted and selectively removed, and fields to be added, rearranged, and removed.

Flexibility, extensibility, and programmability are realized by providing programmers access to the objects which ED itself manipulates. All aspects of ED's appearance and operation can be controlled by the programmer. ED, a restricted or enhanced form of ED, or any of ED's objects, can be embedded in the programmer's applications with no royalties. Objects such as dynamic arrays, windows, data entry windows, macros, menus, browsers, popup directories, and regular expressions are manipulated through normal C function calls.

ED • NUMBER OF BUFFERS LIMITED ONLY BY AVAILABLE RAM • NUMBER OF WINDOWS LIMITED ONLY BY SCREEN SIZE • **POPUP DIRECTORY FACILITY** • VIEW FILESPEC/SUBDIRECTORIES • FIND FILE ON DISK • SEARCH DISK FILES FOR TEXT • COPY • RENAME • DELETE • BROWSE • FILE STATUS • EDIT • EXECUTE • SORT • **SEARCH AND REPLACE** • FORWARD/BACKWARD • CASE SENSITIVE/INSENSITIVE • FULLY PARENTHESIZED REGULAR EXPRESSIONS • INCREMENTAL/GLOBAL SEARCH AND REPLACE ACROSS ALL BUFFERS • **BLOCK COMMANDS** • COLUMN/LINE/STREAM BLOCKS • SAVE • CUT • DELETE • BLANK • SEARCH-FOR • FORMAT • JUSTIFY LEFT/RIGHT/CENTER • UPPER/LOWER CASE • REMOVE • OVERLAY • REACTIVATE • TAB • DRAG • SORTS • SPREADSHEET STYLE MATH - ADD/SUB/MULT/DIVD/AVG • **MACROS** • MENU DRIVEN • VIEW • AUTOEXECUTE • TIME-DELAY • NESTED • RECURSIVE • INTERACTIVE• 1 OR 2 KEYSTROKE • SCREEN DISPLAY CONTROL • SIZE LIMITED BY AVAILABLE MEMORY • AUTOMATIC MACRO MENU CREATION • **MORE FUNCTIONS** • PULLDOWN USER MENU SYSTEM • RECONFIGURE COLOR AND WINDOW STYLE • VIEW/MANIPULATE BUFFERS THROUGH MENU OR KEYSTROKE • INTELLIGENT DIAGRAM/BOX DRAW • ADJUSTABLE SCROLL VALUE • TRANSPOSE LINES/CHARACTERS/WORDS • MOVE-TO-POSITION STACK • VISUAL TAB CREATION • POPUP RULER • COLUMNAR/ENTIRELINE TABS • TIME AND DATE STAMP • RESTORE TYPED-OVER CHARACTERS • CHANGE DIRECTORY • BRACKET/BRACE/PARENTHESIS MATCHING • ENTER GRAPHICS CHARACTERS • EXCUTE DOS COMMANDS • POPUP ASCII TABLE • USER DEFINED POPUP FILES • KEYBOARD CONFIGURATION • PRINT BUFFER WITH PAGE NUMBERS, ADJUSTABLE SPACING AND MARGINS • INTERACTIVE TUTORIAL • MANUAL AND TUTORIAL ON-LINE • WRITTEN IN C AND ASSEMBLER • **EXTERNAL UTILITIES** • INTELLIGENT CHANGE DIRECTORY COMMAND • POPUP DIRECTORY FACILITY • STRING TRANSLATOR • **HARDWARE REQUIREMENTS** • DOS • PC/XT/AT, PS/2, 386 • CGA, MDA, EGA, HERCULES, WYSE 700/AMDEK 1280 • RUNS IN ALL VIDEO MODES, NO FLAGS, NO DRIVERS • 256K • **FOR LITERATURE AND DEMO:**

CALL 201-450-4545

VERSION 1.0

PRICE: $265.00

MASTERCARD, VISA, C.O.D, AND P.O.'S

THE AMERICAN COSMOTRON 80 HOLMES ST • PO BOX 128 • BELLEVILLE, NJ 07109

Number 215 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

FINALLY!

See us at booth R8700
COMDEX/Fall '88
November 14—18, 1988
Riviera Hotel, Las Vegas, Nevada

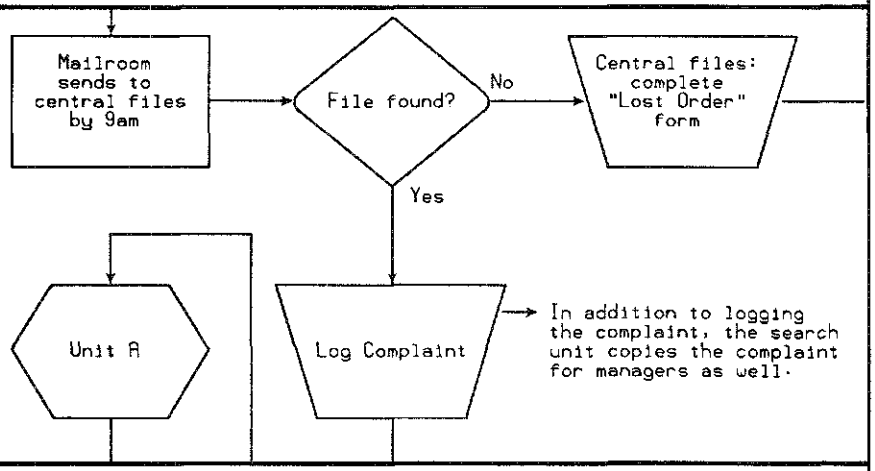
EASYFLOW

An on-screen flowchart processor that knows about flowcharts - not just another "screen draw" program that makes you do most of the work. EASYFLOW is a powerful full-screen graphics program dedicated to flowcharts and organization charts. With it you can quickly compose charts. More important, you can easily modify charts so they are always up to date.

- **Automatic:** Fully automatic text centering within shapes, both horizontally and vertically. Fully automatic line routing & re-routing.
- **Fast:** Written in assembly language for speed.
- **Large:** Charts up to 417 columns wide by 225 lines high. Chart too large for your printer? *EASYFLOW* automatically breaks the chart up & prints it in page size pieces.
- **Standard:** All standard flowcharting shapes included. Custom shapes can be ordered.
- **User friendly:** Don't take our word for it. PC Magazine says *"EASYFLOW lives up to its name. It's hard to imagine any easier and more flexible way to produce basic and even complex flowcharts"*. [March 10th 1987 issue, page 278.]
- **Mouse:** Optional but fully supported.
- **It prints:** On most popular matrix printers including IBM, Epson, Toshiba, HP LaserJet, LaserJet-Plus and many others.
- **It plots:** On HP7440, 7475, 7550, 7570, 7585B and compatible plotters.
- **It works:** We are contractually prevented from mentioning the name of the "big eight" accounting firm that purchased a world-wide site license, but we can tell you that they spent months evaluating all available flowcharting packages before choosing *EASYFLOW*.
- **Rush delivery:** Order by noon today (eastern time) and you'll have it tomorrow. Rush delivery charge is $10.50 (instead of $2.00) and is available only to continental USA and Canada.
- **Documented:** 100 page manual plus over 150 screens of context sensitive help.

EASYFLOW works on IBM PC's, IBM PS/2 and compatibles. Requires 384 K memory, DOS 2.0 or higher and an IBM CGA/EGA/VGA or Hercules monochrome compatible adapter card.

Order direct for only **$149.95** + $2.00 S&H (USA/Canada), $10.00 (foreign). Payment by M.O., cheque, VISA, Mastercard or Company PO.


Mailroom sends to central files by 9am
File found?
No
Central files: complete "Lost Order" form
Yes
Unit A
Log Complaint
In addition to logging the complaint, the search unit copies the complaint for managers as well.

The chart fragment above was produced on an HP LaserJet-Plus and is actual size and unretouched. Publication quality charts like this can be produced using only minutes of preparation time and seconds of print time.

HavenTree Software Limited
PO Box 1093-V
Thousand Island Park, NY 13692

Order Desk: 1-800-267-0668
Information: (613) 544-6035 ext 31
Telefax(G3): (613) 544-9632

Number **154** on the Reader Service Card

in the code file. You might change **CBase** to your name or copyright. Then, if someone tries to remove your notice from the code file, the installation program will no longer work. Sneaky, no?

Improving the Programs

When you run these tests, you'll notice that WINDINST pauses for several seconds while hunting for **CBase** in the compiled code. There's a simple way to reduce this time. First, remove the (* and *) comment brackets from the two **writeln** statements inside TCUNIT's **FoundCBase** function. Recompile WINDINST, and press Q. Jot down the reported offset. When I did this, I got 4768, although your number is likely to be different. The value is decimal, not hex.

Edit TCUNIT.PAS again, and replace the comment brackets in **FoundCBase**. Then change the double IF statement in **ChangesSaved** near the end of the program to read:

```
IF err=0 THEN
 SaveData(f,4768,...)
```

In other words, replace variable **offset** with the value you noted earlier and remove the call to **FoundCBase**. Now when you compile and run WINDINST, the disk update takes place almost instantaneously, as TCUNIT no longer has to hunt for the offset to **CBase**.

Perform this step only after compiling your main program for the last time. (Is there ever a last time?) Any changes to your typed constants or to the program require you to recalculate the offset in the code file. Using the wrong offset is almost sure to lead to a crash.

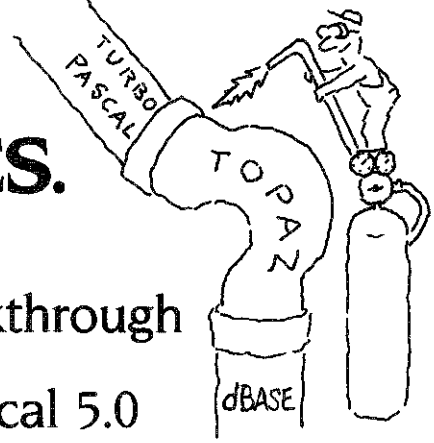
Although these ideas and test programs are experimental, I'm planning to use them as the basis for an installation program in my Turbo Pascal database system as well as in other programs. In the past, I've found many uses for typed constants, but I'm planning to work them in more frequently, especially now that I have a way to demagnetize their previously all-too-static values. ♦

Tom Swan is the author of Mastering Turbo Pascal 2nd Edition *(1988, Howard W. Sams) and other books and is now slaving away on* Mastering Turbo Assembler, *due out early in 1989. Tom is a contributing editor for* Programmer's Journal *and* PC World.

Code follows.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

YOU'LL LOVE THESE UTILITIES.

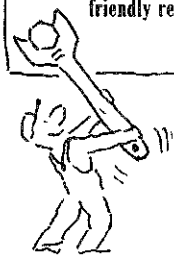


SAYWHAT?!
The lightning-fast screen generator

TOPAZ.
The breakthrough toolkit for Turbo Pascal 5.0

WE GUARANTEE IT!

IRON CLAD MONEY-BACK GUARANTEE.
If you aren't completely delighted with Saywhat or Topaz, return them within 30 days for a prompt, friendly refund.



It doesn't matter which language you program in. With Saywhat, you can build beautiful, elaborate, colorful screens in minutes! That's right. Truly *fantastic* screens for menus, data entry, data display, and help-panels that can all be displayed with as little as one line of code in *any* language. Batch files, too.

With Saywhat, what you see is *exactly* what you get. And response time is snappy and crisp, the way you like it. That means screens pop up instantly, whenever and wherever you want them.

Whether you're a novice programmer longing for simplicity, or a seasoned professional searching for higher productivity, you owe it to yourself to check out Saywhat. For starters, it will let you build your own elegant, moving-bar menus into any screen. (They work like magic in any application, with just one line of code!) You can also combine your screens into extremely powerful screen libraries. And Saywhat's remarkable VIDPOP utility gives all languages running under PC/MS-DOS, a whole new set of flexible screen handling commands. Languages like dBASE, Pascal, BASIC, C, Modula-2, FORTRAN, and COBOL. Saywhat works with all the dBASE compilers, too!

With Saywhat we also include a bunch of terrific utilities, sample screens, sample programs, and outstanding technical support, all at no extra cost. (Comprehensive manual included. Not copy protected. No licensing fee, fully guaranteed). **$49.95**

If you'd like to combine the raw power and speed of Turbo Pascal with the simplicity and elegance of dBASE, Topaz is just what you're looking for. You see, Topaz (our brand new collection of units for Turbo Pascal 5.0) was specially created to let you enjoy the best of *both* worlds. The result? You create truly dazzling applications fast. And no wonder. Topaz is a comprehensive toolkit of dBASE-like commands and functions, designed to help you easily write outstanding, polished programs. Now you can write in Pascal using **SAYs** and **GETs, PICTURE** and **RANGE** clauses, then **SELECT** and **USE** databases and **SKIP** through records, plus **BROWSE, INDEX ON, FIND, PACK, APPEND,** a complete set of time and date handling routines and lots more.

In fact, we've emulated nearly one hundred *actual* dBASE *commands and functions*, and even added *new* commands and functions! All *you* do is declare Topaz's units in your source code and you're up and running!

The bottom line? Topaz makes writing sophisticated Pascal applications a snap. Data entry and data base applications come together with a minimum of code and they'll always be easy to read and maintain.

Topaz comes with a free code generator that automatically writes all the Pascal code you need to maintain a dBASE file with full-screen editing. Plus truly outstanding technical support, at no extra cost. (Comprehensive manual included. Not copy protected. No licensing fee, fully guaranteed). **$49.95**

ORDER NOW. YOU RISK NOTHING. Thousands of satisfied users have already ordered from us. Why not call toll-free, right now and put Saywhat and Topaz to the test yourself? They're fully guaranteed. You don't risk a penny.

SPECIAL LIMITED-TIME OFFER! Buy Saywhat?! and Topaz together for just $85 (plus $5 shipping & handling). That's a savings of almost $15.

To order: Call toll-free
800-468-9273
In California: **800-231-7849**
International: 415-467-6840

The Research Group
100 Valley Drive
Brisbane, CA 94005

☑ **YES.** I want to try:

Saywhat?! your lightning-fast screen generator, so send _____ copies ($49.95 each, plus $5 shipping & handling) subject to your iron-clad money-back guarantee.

Topaz, your programmer's toolkit for Turbo Pascal 5.0, so send _____ copies ($49.95 each, plus $5 shipping & handling) subject to your iron-clad money-back guarantee.

☐ **YES.** I want to take advantage of your special offer! Send me _____ copies of both Saywhat?! and Topaz at $85 per pair (plus $5 shipping & handling). That's a savings of almost $15.

NAME _____
ADDRESS _____
CITY _____ STATE _____ ZIP _____
☐ Check enclosed ☐ Ship C.O.D. ☐ Credit card
_____ Exp. date _____ Signature _____

THE RESEARCH GROUP

Number **198** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

All code in this issue is available on 360K, MS-DOS disk.
ORDER BY PHONE — VISA/MC orders call toll free 800/234-0386.
Mail order information on pages 8 and 88.

Code from "Static Variables" by Tom Swan

1

Listing 1: WINDGLOB.PAS

```
{ WindTest, WindInst typed constants (TC) }

UNIT WindGlob;


INTERFACE

USES  Crt;

CONST

   CBase : String[7] = '@CBASE@';      { Mark beginning of TC area }
{----- Start of default typed-constants area }
   WBForeColor : Word = Blue;          { Border foreground color }
   WBBackColor : Word = LightGray;     { Border background color }
   WTForeColor : Word = Yellow;        { Text foreground color }
   WTBackColor : Word = Blue;          { Text background color }
   WTitle : String[40] = ' Test Window ';  { Window title }
{----- End of default typed-constants area }
   EBase : Char = '!';                 { Mark end of TC area }


IMPLEMENTATION

{ No code in this unit }
END.
```

Listing 2: WINDTEST.PAS

```
{ WindTest application in Turbo Pascal 4.0/5.0 by Tom Swan }

PROGRAM WindTest;


USES  Crt, WindGlob;


PROCEDURE WriteAt( x, y : Word; ch : Char );

{ Display character ch at text display location (x,y) }

BEGIN
   GotoXY( x, y );
   Write( ch )
END; { WriteAt }


PROCEDURE DrawBox( left, top, right, bottom : Word );

{ Outline a box on the text display }

VAR   i : Integer;      { FOR-loop control variable }

BEGIN
   WriteAt( left, top, #213 );
   WriteAt( right, top, #184 );
   WriteAt( left, bottom, #192 );
   WriteAt( right, bottom, #217 );
   FOR i := left+1 TO right-1 DO
```

2

```
   BEGIN
      WriteAt( i, top, #205 );
      WriteAt( i, bottom, #196 )
   END; { for }
   FOR i := top+1 TO bottom-1 DO
   BEGIN
      WriteAt( left, i, #179 );
      WriteAt( right, i, #179 )
   END { for }
END; { DrawBox }


PROCEDURE CenterTitle( s : String; left, top, right : Word );

{ Center string s at row = top between left and right }

VAR   k : Integer;   { Spaces to offset title from left }

BEGIN
   k := ( ( ( right - left ) + 1 ) - Length( s ) ) DIV 2;
   GotoXY( left + k, top );
   Write( s )
END; { CenterTitle }


PROCEDURE OpenWindow( left, top, right, bottom : Word; title : String );

{ Display blank window, restricting subsequent text output to within
the window's borders. }

BEGIN
   TextColor( WBForeColor );
   TextBackground( WBBackColor );
   DrawBox( left, top, right, bottom );
   CenterTitle( title, left, top, right );
   Window( left+1, top+1, right-1, bottom-1 )
END; { OpenWindow }


BEGIN
   ClrScr;
   OpenWindow( 10, 3, 50, 12, WTitle );
   TextColor( WTForeColor );
   TextBackground( WTBackColor );
   ClrScr;
   WHILE NOT Keypressed DO
   BEGIN
      Delay( 15 );  { "Slow down, you move too fast..." }
      Write( Chr( 32 + Random(144) ) )
   END; { while }
   Window( 1, 1, 80, 25 );
   GotoXY( 1, 25 )
END.
```

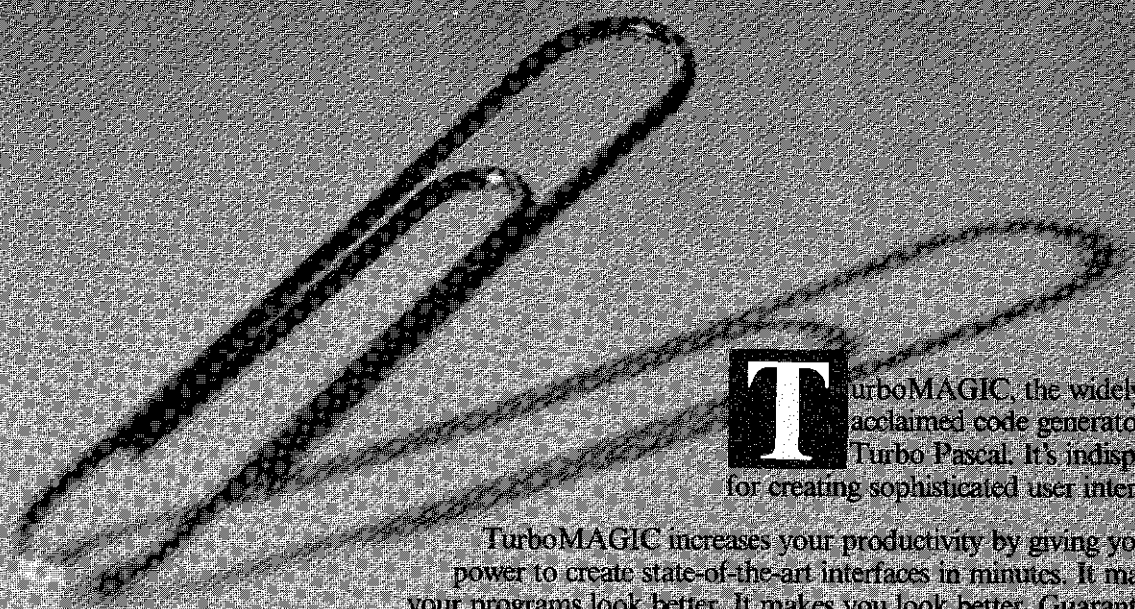
Listing 3: WININST.PAS

```
{ WindTest installer in Turbo Pascal 4.0/5.0 by Tom Swan }

PROGRAM WindInst;


USES  Crt, TCUnit, WindGlob;

CONST FileName = 'WINDTEST.EXE';    { File to modify }

VAR   userQuits : Boolean;          { True to end program }
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Right Tool.



TurboMAGIC, the widely acclaimed code generator for Turbo Pascal. It's indispensable for creating sophisticated user interfaces.

TurboMAGIC increases your productivity by giving you the power to create state-of-the-art interfaces in minutes. It makes your programs look better. It makes you look better. Guaranteed!

Use the highly-praised form painter to design even the most sophisticated input forms. And use the new interactive menu editor to create the menus you need to tie the pieces together. You can't beat it.

You need power. You need speed. You need flexibility. And above all you need a consistent, logical structure throughout your program. ... You need the **Right** tool — turboMAGIC.

Menu Features

- ✓ Borland-style menus
- ✓ Nested to 10 levels
- ✓ Multiple columns
- ✓ Paging/scrolling
- ✓ Disabled entries
- ✓ Pull-down systems
- ✓ Lotus-style menus
- ✓ Options menus
- ✓ Dynamic menus

General Features

- ✓ Pop-up windows
- ✓ Fast screens
- ✓ Mouse interface
- ✓ User color installation
- ✓ Monitor-based colors
- ✓ Directory search windows
- ✓ Context-sensitive & indexed help windows
- ✓ Definable keys
- ✓ Well-documented, easy-to-modify code

Input Form Features

- ✓ Up to 88 lines
- ✓ Scrolling in windows
- ✓ 26 user-modifiable field types
- ✓ Format masks
- ✓ Conditional-use fields
- ✓ Fast automatic updating of dependent fields
- ✓ Bomb-proof input handlers
- ✓ Output to printer or disk

"... the best screen generator on the market." George Kwasha, TUG Lines, Nov/Dec 87

"... the Cadillac of prototyping tools for Turbo Pascal." Jim Powell, Computer Language, Jun 87

ORDER your Magic TODAY! Only $199.
CALL TOLL FREE 800-225-3165 or 205-342-7026

turbo MAGIC

Version 2.0

Sophisticated Software Inc., 6586 Old Shell Road, Mobile, AL 36608

Requires 512K IBM PC compatible and Turbo Pascal 4.0 or 5.0. 30-Day Money Back Guarantee. S&H: US $5, Canada $15, Other $25

Number 257 on the Reader Service Card.

PRODUCED BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Swan Code, Continued

3

```
FUNCTION MenuChar : Char;

{ Display menu and return selection }

BEGIN
   ClrScr;
   Writeln( 'WindTest Installation Program' );
   Writeln( '-------------------------------------' );
   Writeln( 'A - Border foreground ..... ', WBForeColor:2 );
   Writeln( 'B - Border background ..... ', WBBackColor:2 );
   Writeln( 'C - Text foreground ....... ', WTForeColor:2 );
   Writeln( 'D - Text background ....... ', WTBackColor:2 );
   Writeln( 'E - Window title .......... ', WTitle );
   Writeln;
   Writeln( 'Q - Quit and save changes' );
   Writeln( 'X - Quit and do NOT save changes' );
   Writeln;
   Write(   'Pick a letter, any letter: ' );
   MenuChar := UpCase( ReadKey )
END; { MenuChar }


{$V-}    { Turn off string-parameter length checking }

BEGIN
   userQuits := False;
   REPEAT
      CASE MenuChar OF
         'A' : GetWord( 'Border foreground', WBForeColor, 0, 15 );
         'B' : GetWord( 'Border background', WBBackColor, 0, 15 );
         'C' : GetWord( 'Text foreground', WTForeColor, 0, 15 );
         'D' : GetWord( 'Text background', WTBackColor, 0, 15 );
         'E' : GetStr( 'Window title', WTitle, 40 );
         'Q' : userQuits := ChangesSaved( fileName, CBase,
                                 Ofs( CBase ), Ofs( EBase ) );
         'X' : userQuits := True
      END { case }
   UNTIL userQuits;
   GotoXY( 1, 25 )
END.
```

Listing 4: TCUNIT.PAS

```
(*
 *    PROGRAM : For editing typed constants in EXE files
 *    SYSTEM  : PC/MS-DOS / Turbo Pascal 4.0 or 5.0
 *    AUTHOR  : © 1988 by Tom Swan
 *)

UNIT TCUnit;


INTERFACE

USES  Crt;     { Standard Borland display unit }

PROCEDURE GetWord( prompt : String; VAR v : Word; low, high : Word );

PROCEDURE GetStr( prompt : String; VAR s : String; maxLen : Word );

FUNCTION ChangesSaved( fileName, searchStr : String;
   cbase, ebase : Word ) : Boolean;
```

4

```
IMPLEMENTATION


TYPE  ExeFile = File OF Char;     { Reads EXE files as file of char }


PROCEDURE GetWord( prompt : String; VAR v : Word; low, high : Word );

{ Prompt for word value, displaying prompt string and limiting
response in v to the range low..high. }

VAR   response : String[8];       { Holds response to prompt }
      newValue : Word;            { Possible new value for v }
      e : Integer;                { Error code for Val() }

BEGIN
   ClrScr;
   Writeln;
   Writeln( prompt, ' = ', v );
   Writeln;
   Writeln( 'Enter new value from ', low, ' to ', high );
   Write(   'or press Enter for no change: ' );
   Readln( response );
   IF Length( response ) > 0 THEN
   BEGIN
      Val( response, newValue, e );
      IF ( e = 0 ) AND ( low <= newValue ) AND ( newValue <= high )
       THEN
         v := newValue
       ELSE
         BEGIN
            Writeln;
            Write( 'Entry error.  Press Enter...' );
            Readln
         END { else }
   END { if }
END; { GetWord }


PROCEDURE GetStr( prompt : String; VAR s : String; maxLen : Word );

{ Prompt for string, displaying prompt string and limiting response
in s to string length 0..maxLen. }

VAR   response : String;       { Holds response to prompt }

BEGIN
   ClrScr;
   Writeln;
   Writeln( prompt, ' = ', s );
   Writeln;
   Writeln( 'Enter new string with up to ', maxLen, ' characters' );
   Write(   'or press Enter for no change: ' );
   Readln( response );
   IF Length( response ) > 0 THEN
   BEGIN
      IF Length( response ) <= maxLen
       THEN
         s := response
       ELSE
         BEGIN
            Writeln;
            Write( 'Entry error.  Press Enter...' );
            Readln
         END { else }
   END { if }
END; { GetStr }
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MEMO:

Take a break from your programming. Relax with an interactive fiction MegAdventure from MISOSYS. Adventures sharpen your wits, and keep you on your toes. MegAdventures challenge you like no other.

Pit your wits against an ill-tempered dragon. Track it down in the mysterious northern mountains, the likes of which only heart-quickening tales of terror and death are told? Lair of the Dragon contains well over 160 locations, over 200 recognizable objects, and has a vocabulary of over 600 words.

Lair of the Dragon [M-86-021]: Just $29.95
S&H: $2 US, $3 Foreign. MC & VISA.

MISOSYS, Inc.
P.O. Box 239
Sterling, VA 22170-0239
800-MISOSYS or 703-450-4181

Number 238 on the Reader Service Card

dp_MAX

dBase III Tools in Turbo Pascal 4.0

Complete Support for dBase III files.
DBF, NDX, DBT file & record Access.
Fully Compatible dBase III B+Tree ISAM.
Library of 100+ functions in TP4 Unit.
Allows 250+ files & indexes open at once.
LRU file caching, round robin file manager.

$$dp\_MAX = \int_{-\infty}^{+\infty} \frac{dBASE\ III\ +}{Turbo\ Pascal\ 4.0}$$

Max Software Consultants, Inc.
4101 Greenmount Avenue
Baltimore, MD 21218
(301)-323-5996

$149

Number 213 on the Reader Service Card

MISOSYS

Programmer's Journal downloads are now available from MISOSYS' forum on Compuserve: GO PCS49

DED86™ Version 2

Powerful features in Version 2!

When you need to travel through your disk drive, why settle for a tool that isolates you from the road? DED86 gives you the direct controls you need to explore your disk. It's a full-screen sector-oriented disk/file editor and a page-oriented memory editor. When you want to "unerase" erased files, DED86's flexible KEEP facility does the job without you fussing over FATs.

- Look by cylinder/head/sector, sector or cluster
- Scan free clusters to search for erased data
- Jump about subdirectories
- Keep sectors & clusters for writing to a file
- Edit bytes in hexadecimal or ASCII, zap in 0s
- Search disk, file, or memory in ASCII or hex
- Touch directory with your date and/or time
- Obtain complete disk statistics in one screen
- Alter attributes: archive, system, hidden, read
- Save/Restore sectors to/from auxiliary buffers
- DOS subshell available while using DED86
- Handles 5.25" & 3.5" drives & RAM disks

RATFOR-86™ 2.0

If you're still fussing with FORTRAN, RATFOR-86 by James Beard, Ph.D., provides structure and greater portability to FORTRAN programs; reduce your programming time and effort dramatically over that required when writing directly in FORTRAN.

- Has extensive macro support
- "arith" macro for binary arithmetic
- 6 new conditional macros in 2.0
- short form READ and PRINT
- keyword "cswitch" like in C
- Requires a FORTRAN compiler

ED/ASM-86™

Integrated Assembler

Editor, assembler, linker, and debugger are integrated into a single .EXE file to ease you into assembly language. Handles 8086, 80186, 80286, and 8087.

- A full featured line EDITOR with intra-line editing, block move/copy, partial save/load.
- A full-screen "Turbo-type" editor included
- ASSEMBLER supports structured coding, can directly generate .COM or .EXE file; assemble to memory for faster debugging.
- The DEBUGGER supports symbolic disassembly of arbitrary code to disk

Purchase now and get a free update to **ED/ASM-386** available soon.

DED86 . . . . . $59.95
ED/ASM-86 . . . . . $99.95
RATFOR-86 . . . . . $99.95
M/C and VISA accepted
S&H: $5, $6 Canada, $15 Other

MISOSYS, Inc.
P.O. Box 239
Sterling, VA 22170-0239
800-MISOSYS or 703-450-4181

Number 121 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Swan Code, Continued

5

```
PROCEDURE ShowError( e : Integer );

{ Display an error message.  e>0 = I/O error; e<0 = other error }

BEGIN
   IF e > 0
      THEN Writeln( 'I/O Error #', e )
      ELSE Writeln( 'Error in EXE file format' );
   Writeln;
   Writeln( 'WARNING: EXE file may be damaged!' );
   Writeln;
   Write( 'Press Enter...' );
   Readln
END; { ShowError }


{$i-}    { Shut off I/O error checks }


FUNCTION FoundCBase( VAR f : ExeFile; VAR searchStr : String;
   VAR offset : LongInt ) : Boolean;

{ Return True if searchString (CBase) is found in file f.  If found,
then return byte offset to string in file. }

VAR   position : LongInt;        { Possible position of match }
      ch : Char;                 { Holds candidate bytes from file }

   FUNCTION FoundMatch : Boolean;
   { True if current position = search string }
   VAR   i : Integer;   { searchStr index }
   BEGIN
      FOR i := 2 TO Length( searchStr ) DO
      BEGIN
         Read( f, ch );
         IF ch <> searchStr[i] THEN
         BEGIN
            FoundMatch := False;
            Exit
         END { if }
      END; { for }
      FoundMatch := True
   END; { FoundMatch }

BEGIN
   Reset( f );              { Start search at beginning of file }
   WHILE NOT Eof( f ) DO
   BEGIN
      Read( f, ch );
      IF ch = searchStr[1] THEN        { Test one char }
      BEGIN
         position := FilePos( f );     { Remember position }
         IF FoundMatch THEN            { Check for match }
         BEGIN
            offset := position - 2;    { Found: return offset }
            (* Writeln; Writeln( 'Offset = ', offset ); *)
            FoundCBase := True;        { Set function result }
            Exit                       { Stop searching }
         END ELSE
            Seek( f, position )        { Continue search }
      END { if }
   END; { while }
   FoundCBase := False     { searchStr isn't there }
END; { FoundCBase }
```

6

```
PROCEDURE SaveData( VAR f : ExeFile; offset, cbase, len : LongInt;
  VAR e : Integer );

{ Write len bytes to file f, beginning at byte #offset in the file
and transferring from memory len bytes starting at DS:cbase.  Return
any errors in e.  This copies the in-memory typed constants to the
EXE file image on disk. }

VAR   i : Word;       { Mem[] array index }

BEGIN
   Seek( f, offset );
   FOR i := 0 TO ( len - 1 ) DO
   BEGIN
      Write( f, Char( Mem[ DSeg:( cbase + i ) ] ) );
      e := IoResult;
      IF e <> 0 THEN Exit
   END { for }
END; { SaveData }


FUNCTION ChangesSaved( fileName, searchStr : String; cbase,
   ebase : Word ) : Boolean;

{ Return True if typed constants in memory are written to disk.
fileName must be a Turbo Pascal compiled EXE file.  searchStr
should equal the CBase marker string at the start of the typed
constants area.  cbase should be the in-memory offset to the CBase
typed constant.  ebase should be the in-memory offset to the EBase
typed constant. }

VAR   f : ExeFile;          { Read .EXE as a char file }
      offset : LongInt;     { Byte offset to CBase in .EXE file }
      err : Integer;        { Error code }

BEGIN
   GotoXY( 1, 25 );
   ClrEol;
   Write( 'Saving changes to ', fileName, '. Please wait...' );
   Assign( f, fileName );
   Reset( f );
   err := IoResult;
   IF err = 0 THEN
      IF FoundCBase( f, searchStr, offset )
         THEN SaveData( f, offset, cbase, ( ebase - cbase ), err );
   ChangesSaved := ( err = 0 );  { i.e. True if no error }
   Writeln;
   IF err = 0
      THEN Writeln( 'Changes saved' )
      ELSE ShowError( err );
   Close( f )
END; { ChangesSaved }


END. { TCUnit }
```

♦

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Heap Expander™ version 2.5

Now your programs can have virtually *unlimited* heap space using expanded memory, extended memory, disk space, or any combination of the three. And it's all transparent. The Heap Expander's initialization code checks the system's resources and uses whatever is available.

only $79.95*

- Uses LIM-standard expanded memory if present.
- Uses AT-style extended memory if present.
- Swaps data to disk as needed.

Libraries and Source Code for:

- Turbo C
- Microsoft C 4.0 and 5.0
- Turbo Pascal 3.0 and 4.0

Requires an IBM PC, XT, AT, or close compatible with MS-DOS or PC-DOS version 2.0 or above

MC/VISA/COD call
1-800-248-1045 x 100 (US)
1-800-952-5560 x 100 (Idaho)

*CA residents add 6 1/2% sales tax. Foreign customers, please add $4.00 for shipping and handling.

The Tool Makers

75 Mt. Hermon Road
Scotts Valley, CA 95066
(408) 458-0690

Number **166** on the Reader Service Card

People are talking about us.

"This is a very good implementation of FORTRAN, better than the one I use on my minicomputer system." *IEEE Software*

"Lahey F77L is definitely for the programmer's market, with features for the casual and professional user...*EDITOR'S CHOICE" PC Magazine*

"It is a fantastic product." *PC Australia*

"It should be the last FORTRAN compiler you will need to buy..." *Your Computer*

"Lahey F77L was by far the easiest compiler to use..." *Programmer's Journal*

"Compilation speed is Lahey FORTRAN's most remarkable feature; it is unbelievably fast...Realistically, Lahey FORTRAN is the most efficient and productive FORTRAN development tool for the DOS environment..." *Byte*

When people talk about FORTRAN the name mentioned most often is Lahey Computer Systems Inc.

Contact us to discuss our products and your needs. (800) 548-4778
Lahey Computer Systems, Inc. P.O. Box 6091, Incline Village, NV 89450
Tel: (702) 831-2500 FAX: (702) 831-8123 Tlx: 9102401256

Number **44** on the Reader Service Card

DOS & OS/2
Free Demo Diskette Available

Q. How many programmers does it take to maintain a MAKE dependency file?

A. NONE! *If you use* VersiMAKE™

VersiMAKE™ is a full-featured MAKE *utility that includes:*

■ **Dependency Generation**
Derives your system's dependencies, through analysis of its C and MASM source files. Say goodbye to manual maintenance of MAKE dependency files!

■ **Wild Card File Name Matching**
Analyzes an *entire collection of source* files with a single statement.

■ **Nested Include Files**
Handles standard C and MASM "include" conventions, and the INCLUDE environment variable.

■ **Powerful Macro Facilities**
Built-in macros, user-defined macros, and environment variables.

■ **Analytical Reports**
Shows the entire Include file hierarchy for each source file analyzed, and all of the parent source files for each include file.

Q. How many programmers does it take to trace a symbol thru your system?

A. ONE! *If you use* VersiCREF™

VersiCREF™ is a unique utility that creates a Master Cross-Reference of your entire system.

■ **Multi-Lingual**
Handles C, assembler, or both.

■ **Flexible**
File names with line numbers, or file names alone. Global and local symbols, or globals alone.

■ **Powerful**
Easily handles systems containing 100 source files or more.

VersiMAKE™	$125
VersiCREF™	$75
Both	$150

(Outside U.S., add $5 for shipping and handling)

800-334-4096
(In NJ, 609-871-0202)
MC/VISA/AMEX accepted

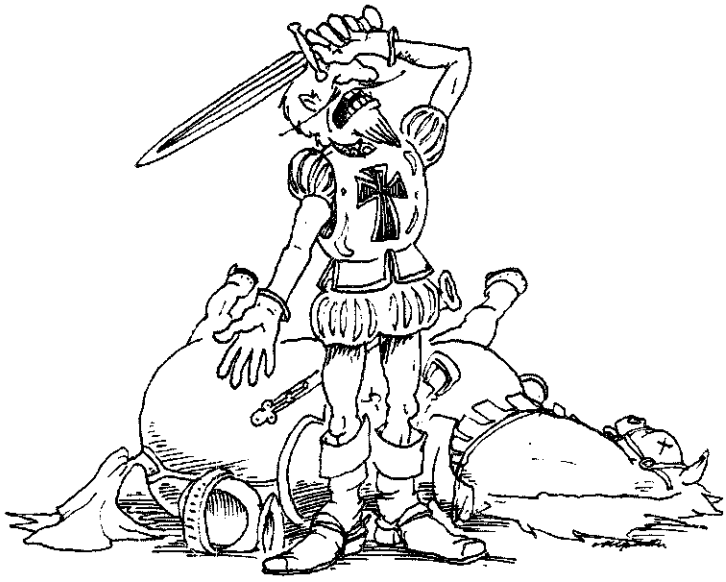
SUMMIT INFORMATION SYSTEMS, INC.
73 East Lane, Willingboro, NJ 08046

Number **158** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

My Kingdom for an Overlay!

by
Bill Madison



The ability to overlay large applications is taken for granted by most modern software developers. Thus, the users of Borland International's Turbo Pascal were quite shocked to discover that a built-in overlay capability present in Release 3 was missing from Release 4.

The trauma induced by this lack was alleviated when TurboPower Software provided an overlay package. Using a post-processor, the software processed a TP4 .EXE file into a new .EXE file plus an .OVR file. The total cure came with the advent of Turbo Pascal Release 5—Borland's reintroduction of an embedded overlay capability.

Not surprisingly, the program designer must use entirely different setup procedures when using the TurboPower Software system versus the Borland system. How do the two schemes compare? What application design and implementation questions must be resolved in using each system?

Ultimately, these issues resolve into a single question: "Shall I stick with TP4 or upgrade to TP5 for both current and future work?" TurboPower Software's overlay manager cannot work with TP5 except under very restrictive circumstances. Further, I have been informed that no current plans exist to port the TurboPower product over to TP5 and place it in competition with the Borland overlay manager.

I have had an excellent opportunity to examine this question, since I am currently working on two quite different overlaid applications. One is my Shareware product, ReadyRef. ReadyRef is an overlaid TSR, primarily I/O bound. Thus, the overriding design problem (and the only reason for using overlays) is keeping resident portion size as small as possible consistent with reasonable response time.

The other application—a large, very complex transportation simulation package—is heavily compute bound, written for a client. The primary concern here is minimizing overlay thrashing, which enhances product maintainability.

Both products have evolved from a rather simple, nonoverlaid design into an overlaid application as new features have been incorporated. Thus, I have been able to examine problems associated with:

- Conversion of a nonoverlaid application to an overlaid application using the TurboPower Software overlay manager (ReadyRef)
- Conversion of an application using the TurboPower Software overlay manager to one using the Borland overlay manager (ReadyRef again)
- Conversion of a nonoverlaid application to an overlaid application using the Borland overlay manager (the simulation product)

The remainder of this article compares the structure and operation of these two overlay managers. For simplicity's (and brevity's) sake, let's agree to refer to TurboPower Software as "TPS," and Borland International as "BI."

Structure and Operation of the Overlay Managers

Both the TPS and the BI overlay managers have unit-level granularity. This is unlike the overlay manager embedded in TP3, which had a granularity at the block (i.e., procedure or function) level. No particular problem arises as long as you stay aware of this. Also in contrast with the TP3 system, both the TPS and the BI systems produce only a single overlay file (with an extension of .OVR), which contains all of the overlaid units.

With this much said, however, any similarity between the two ends. In the following section, I will describe various differing attributes of both systems in parallel and score each (in my usual impeccably objective manner) on a scale of 1 to 5 (1 = low).

What Can and Cannot Be Overlaid, and Other Cautionary Notes

The following should not be overlaid within TPS:

- The OVERMGR (overlay manager) unit itself (aren't you surprised?)
- Units containing interrupt handlers
- Units with routines called directly or indirectly via procedure pointers

Additionally, care must be taken when passing code segment–based constants as actual parameters if the caller and the block being called are in different units that share the same overlay group. (An *overlay group*, a concept unique to the

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

TPS system, is a designer-designated collection of units that share the same space in the overlay buffer. There can be many overlay groups in a single application.)

The problem with code segment–based constants is circumvented easily by forcing the argument onto the stack before calling the target block, as in the following example:

Assume that unit X contains procedure X1 and uses unit Y. Further assume that unit Y contains procedure Y1, and that both units share the same overlay group. Finally, assume that procedure Y1 takes two VALUE parameters, S : String and I : Integer.

Then, if procedure X1 contains the following statements, erroneous results will occur:

```
const I = 100;
...
Y1('Hi, there!', I);
...
```

The problem may be easily remedied by changing the call to force the parameters onto the stack. This can be accomplished (take my word for it) by placing the constants in expressions that must be evaluated immediately prior to transferring control to the target block:

```
: const I = 100;
...
Y1('Hi,there!'+'', I+0);
...
```

The following should not be overlaid within BI:

- The OVERLAY (overlay manager) unit itself
- Units containing interrupt handlers
- Units not compiled with the $O+ compiler directive

Score: TPS - 2, BI - 5

How Many Can Be Overlaid?

As delivered, the TPS system can support up to six overlay groups, each of which can contain a maximum of 51 units. The entire program can also have a maximum of 51 overlaid units. Of course, additional nonoverlaid units can be used, but the total (overlaid + nonoverlaid) cannot exceed 255 units.

These limits can be changed easily; required steps are well documented. Two constants must be changed in an Include file, and identically named constants must be changed in an .ASM file. The .ASM file must then be assembled to an .OBJ file. Finally, both the OVRMGR unit and the OVERLAY program must be rebuilt. Additionally, the application to be overlaid must first be compiled to a vanilla-flavored .EXE file and then post-processed into the final .EXE and .OVR files. The total size of the application is limited by the maximum-size .EXE file that TP4 can produce (approximately 1 megabyte).

No limitations are stated in the BI documentation.

Score: TPS - 4, BI - 5

Designing the Overlay Structure

Within TPS, specifying the overlay structure requires only the creation of one text file (with an .OSP extension) in addition to the normal program files. This file contains—on each line—the name of a unit to be overlaid and the designation of the overlay group into which it is to be placed.

Only one program change is required to use the overlay system. This is the addition of the OverMgr unit at the head of the Uses clause in the main program.

The decision as to which units to overlay must be determined on the basis of the programmer's knowledge of the application. No deterministic algorithm is provided.

A similar decision is required for assignment of an overlay group. However, some aids are available to make the job a good deal easier than pure hit or miss.

First, when compiling the program, it is necessary to create a .TPM file and convert it to a .MAP file (by running the TpMap program). The .MAP file, in conjunction with the .OSP file, is used as input to the OVERLAY program to create the new .EXE and .OVR files.

The existence of a .MAP file correctly reflecting the sizes of the units making up the program is additionally useful. The file directs the designer's attention to those units that might benefit from redesign or fragmentation.

Second, the creation of the final .EXE and .OVR files is the result of execution of the post-processor OVERLAY program against the original .EXE file. Thus, you can try a new overlay structure without recompiling the application: Just modify the .OSP file and rerun the post-processor.

Considering both factors, what could be a seat-of-the-pants guessing game can become a series of controlled experiments. To my mind, this is one of the most significant advantages of the TPS system.

Several steps are required in using the BI overlay system:

1. All units used by the application must be mentioned in the Uses clause of the main program. (I find this a definite minus vis-à-vis program readability. I strongly prefer to mention those units actually used by a program or unit.)
2. The source file for the main program must be edited to include the Overlay unit in its Uses clause, ahead of the first unit to be overlaid.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

3. This source file must be further edited by including a {$O <unitname>} compiler directive for each unit to be overlaid.
4. All units that are to be overlaid must be compiled with the $O+ compiler directive. (The effect on code size here seems to be negligible, adding only one or two paragraphs at most to the size of the .TPU file. The largest effect I have seen is ten paragraphs for a general utility unit of mine that contained several dozen blocks.)
5. All blocks that directly or indirectly call a block in an overlaid unit must be compiled using a FAR call. In effect, this means that every unit and the main program had better be compiled with the $F+ compiler directive. The only exception is nonoverlaid units that are completely self-contained.

Score: TPS - 5, BI - 2

Overlay System Initialization

Under normal conditions, overlay-system initialization occurs automatically in TPS and need not concern the programmer. The primary exception to this occurs when writing TSRs. In this case, a call to the procedure SetOverlayHandler should occur each time the TSR is activated. (There are certain other processes not directly concerning the overlay system—but affecting it—that should be performed at the same time. These are well documented.)

Initialization of the overlay system in BI, except under certain rather sterile (for me) conditions, represents an esoteric combination of diabolism and voodoo. The TP5 Reference Guide stresses the importance of including the Overlay unit in the Uses clause of the main program, ahead of the first overlaid unit. The guide also mentions the need to force initialization of the overlay unit prior to execution of any code (including initialization code) for any overlaid unit. This may mean that the calls to OvrInit, OvrSetBuf, and OvrInitEMS must be placed in the initialization section of a separate, nonoverlaid unit. In addition, the calls must appear in the Uses clause of the main program, ahead of the mention of the first overlaid unit.

Nowhere in the guide could I find (and it took me four transcontinental phone calls to BI to discover) that the same rules apply when any unit—overlaid or not—contains initialization code that manipulates the heap in any way. Again, overlay-system initialization must precede execution of the heap-diddling initialization code of the guilty unit. This can be frustrating particularly when you do not have access to a unit's source code. How do you know?

The TP5 Reference Manual is, in my humble opinion, downright misleading in this regard. In the discussion of the OvrSetBuf procedure (pp. 344ff) is the statement: "OvrSetBuf requires that the heap be empty;. . . make sure to call OvrSetBuf before the Graph unit's InitGraph procedure. . . ." I find it difficult to determine how a reasonable person (among this collection I clearly include myself) could infer from that statement that the Graph unit includes heap-diddling initialization code.

Score: TPS - 5, BI - 2

Use of EMS

The TPS overlay system automatically uses EMS (Lotus/Intel/Microsoft Expanded Memory Specification) memory if it is available. To prohibit the use of EMS, change one constant in a defined "user patch area," and recompile the overlay manager unit. Alternatively, you may use an installation program to set this and other constants. (Source is provided for this installation program to customize it for a particular application.)

Since the EMS usage decision is made at execution time rather than at link time, all of the code relating to EMS will remain in the .EXE file. Should you need to reduce to a minimum the nonoverlaid portion size of the application, you must excise the EMS-related code from three files: the .PAS files for the overlay manager unit and the overlay post-processor, and the .ASM file common to both. All of the code is extremely well commented, however, making such an excision quite straightforward. (It worked for me on the first try—truly remarkable considering my fat fingers and the not-always-crystal clarity of my mental processes.)

Use or nonuse of EMS within the BI system is a model of simplicity. Following the initialization of the overlay system, but prior to any other function, a call to the procedure OvrInitEMS causes the overlay system to use EMS if it is present. If it is not present, the overlays will be loaded from disk as required.

Score: TPS - 3, BI - 5

Thrashing

The TPS system has retained the functionality of the TP3 separate overlay files by permitting the user to define separate "overlay groups." Blocks within one overlay group can call blocks within another overlay group without causing thrashing.

The default state of the BI system will not allow thrashing to occur if the sum of the sizes of the units involved is less than the size of the largest unit being overlaid. If this condition is not met, thrashing will occur. However, the BI system provides a mechanism for overcoming this problem by using the procedure OvrSetBuf. This procedure accepts a single-value parameter, which is the new size for the overlay buffer. If the buffer size is set at the size required by the maximum sum of the sizes of all units (which must be co-resident in order to satisfy stacked block calls), then unnecessary thrashing will be eliminated.

The BI system does make it difficult to determine just what this number is to be. Perhaps the simplest way to explain this problem is by contrasting the two systems.

The TPS system creates the .OVR file using a post-processor program that operates on the compiler-generated .EXE file. When you generate a .MAP file as part of the compilation, the exact RAM-based size of each unit can be determined. (Aside from any other benefits, this procedure provides an excellent development aid: It points to those units that can be fragmented to reduce materially the application's RAM requirements.) The TPS system then automatically sets the size of the overlay buffer to accommodate the largest unit within the defined overlay groups.

In contrast, the BI system produces both the .EXE and the .OVR files as direct output from the compiler. Therefore, any .MAP file produced will display only a small residuum for each overlaid unit (typically less than 100h bytes.) I have found absolutely no clue as to the actual RAM requirements of an overlaid unit.

Score: TPS - 5, BI - 2

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Celebrating our 10th Anniversary!

AMX™ 68000

KADAK's engineers bring years of practical real-time experience to over 600 installations world-wide.

This *real time* MULTITASKING KERNEL simplifies *real life* product development

- No royalties
- Full source code included
- * C interface
- Preemptive scheduler
- Intertask messages
- Terminal Handler
- * Dynamic memory allocation
- * Event Manager
- * Semaphore Manager
- * Resource Manager
- * Buffer Manager

AMX 68000 operates on any 68000/10/20 system

Manual only $75 US
AMX 68000 $1600 US
* Options — ask for price list.
(Shipping/handling extra)

Also available for 8080, Z80, 8086/186/286.

KADAK Products Ltd.
206 - 1847 W. Broadway
Vancouver, B.C. Canada V6J 1Y5
Telex: 04-55670
Fax: (604) 734-8114
Telephone: (604) 734-2796

Number 211 on the Reader Service Card

New & Improved Version 2.1

TRANSLATOR
(Pascal to C)

- *For Programmers and those learning* **C**
- Obtain the Attributes and Extensions of the Powerful **C** Language from Your Natural Language Pascal Programs
- Port your Pascal Programs to UNIX and other **C** Environments
- Now supports Strings and Pascal I/O
- Convert Entire Programs or Program Segments Interactively.
- User Manual and User Disk Contain Many Illustrative Example TRANSLATIONS with Explanations of Significant Difference between the Pascal and the **C** Programs.

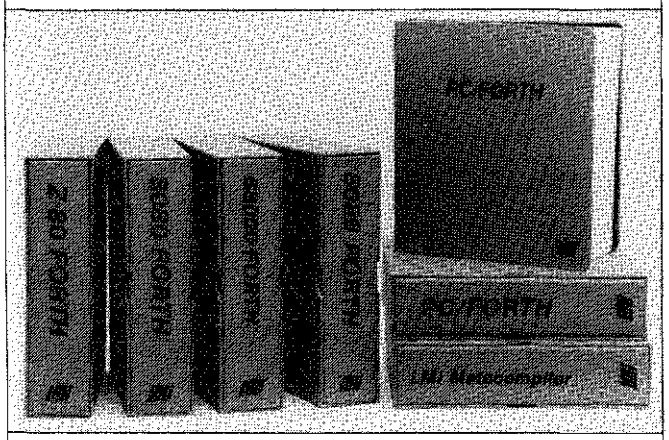
New Low Price Only: $99.95

MILTON BROWN SOFTWARE
935 Chester Avenue
San Marino, CA 91108

(818) 796-6934

Number 240 on the Reader Service Card

TOTAL CONTROL with *LMI FORTH*™



For Programming Professionals:

an expanding family of compatible, high-performance, Forth-83 Standard compilers for microcomputers

For Development: Interactive Forth-83 Interpreter/Compilers

- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 400 page manual written in plain English
- Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support

For Applications: Forth-83 Metacompiler

- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, and 6303
- No license fee or royalty for compiled applications

For Speed: CForth Application Compiler

- Translates "high-level" Forth into in-line, optimized machine code
- Can generate ROMable code

Support Services for registered users:

- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.

Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to: ***(213) 306-7412***

Overseas Distributors.
Germany: Forth-Systeme Angelika Flesch, Titisee-Neustadt, 7651-1665
UK: System Science Ltd., London, 01-248 0962
France: Micro-Sigma S.A.R.L., Paris, (1) 42.65.95.16
Japan: Southern Pacific Ltd., Yokohama, 045-314-9514
Australia: Wave-onic Associates, Wilson, W.A., (09) 451-2946

Number 63 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Developer's Data Manager

AXOS™

It Speaks Your Language!

BASIC • COBOL • C • FORTRAN • PASCAL • ASSEMBLER

A New Standard in Access Method Technology

AXOS is a *fast* full-featured, key-indexed file management system designed to provide application programmers and software developers a flexible and versatile data storage and retrieval mechanism with the least amount of overhead. It can be used in either single or multi-user LAN environments on the IBM PC PS/2 and compatibles. As AXOS can be called from any programming language through a common communication protocol, it allows one to manipulate data in a manner not possible with other DBMSs or access methods. AXOS also offers a full complement of file mangement functions, system management functions, productivity tools and debugging aids designed to ensure easy and accurate development and support of microcomputer applications systems.

FEATURES

- **SECURITY: LAN computer name and password protection. Eight types of file I/O security by computer.**
- **AUDIT: Complete (update and retrieval) or partial (retrieval) capture of all AXOS activity, available with selection criteria both online or hardcopy.**
- **DATA INTEGRITY: Ensures full data integrity in the event of a program, system or power failure.**
- **INTERACTIVE MONITOR: Instantaneous interactive monitoring of AXOS activity as a program is executing.**
- **FILE CONVERSION: Convert or merge existing files of other types into any one of the AXOS file formats available.**
- **FILE LIST: Produces a formatted report of the contents of all AXOS file types and a variety of DOS sequential file types.**
- **PROGRAM INVOKED ERROR ROUTINE: Used by application programs for processing unexpected I/O return codes.**
- **RECORD ENCRYPTION AND COMPRESSION: All records contained on AXOS files are encrypted and compressed so as to reduce disk storage requirements.**
- **PAGE BUFFERING: AXOS builds and manages all of its own buffer space so as it minimize disk I/O and decrease processing time in both single user on multi-user (LAN) environments.**
- **PARAMETERS: All of the AXOS keyboard commands utilize parameters entered at the time that the command is executed. This provides a level of flexibility wherein a command may be tailored to meet specific user requirements.**

SPECIFICATIONS

- **Languages Supported:**

  BASIC COMPILED BASIC COBOL
  FORTRAN C ASSEMBLER PASCAL

- **Local Area Networks Supported:**

  IBM® PC NET AT&T® StarLAN
  3Com® 3+Share Novell® Netware

• File Types Supported .....:	**Sequential Single Index Multi Index (cluster)**
• Maximum Indexes .....:	**64 per file**
• Maximum Key Length .....:	**240 bytes for unique keys 236 for non-unique keys**
• Maximum Record Size .....:	**64,000 bytes**
• Maximum Records Per File .....:	**16,777,216**
• File Capacity .....:	**4.2 billion bytes**
• Maximum Files & Indexes allowed open at one time .....:	**253**
• Operating System Requirement :	**DOS 2.1 or above 128 - 256KB of memory**

Come To Grips With The Power Of AXOS

"McSoft's AXOS is not for the faint of heart - it clearly is aimed at the professional developer who is building end-user applications or intermediate development systems." PC Tech Journal 10/88.

McSoft INCORPORATED™

AMEX/COD/MC/VISA

$595 **Includes 30 days free technical support**
Extended technical support available via HOTLINE

And there are no royalties on your AXOS applications!

FOR MORE INFORMATION OR TO ORDER CALL (404) 237-4169

P.O. Box 12044 • Atlanta, GA 30355-2044

Number **230** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Summary and Conclusions

Weighing the scores equally for the five system attributes examined above results in the following average scores:

ATTRIBUTE	TPS	BI
What can be overlaid:	2	5
How many can be overlaid:	4	5
Application design:	5	2
Initialization:	5	2
Use of EMS:	3	5
Thrashing:	5	2
Average scores:	4.0	3.5

However, I don't consider these attributes to be equal. Instead, I would weigh application design most heavily, followed by initialization. I would then assign median weight to what can be overlaid and use of EMS. Finally, I would assign thrashing a lower-than-average weight. Assessing the scores with these new factors provides the following results:

ATTRIBUTE	W	TPS	BI
What can be overlaid:	2	2 4	5 10
How many can be overlaid:	1	4 4	5 5
Application design:	4	5 20	2 8
Initialization:	3	5 15	2 6
Use of EMS:	2	3 6	5 10
Thrashing:	1	5 5	2 2
Sum of Weights:	13		
Average scores:		4.2	3.2

Assign weights as you will. At least you should now have some basis for judgment.

Last, but certainly not least, we should consider the importance of TP5's other features that are not available in TP4. While not easily quantifiable, certainly the value of these features must be weighed in making the final program selection. This factor must be given even greater weight when designing new applications than when converting applications that have been developed and are operational under TP4, as mine were.

Considering all factors, however, I would tend to use the TPS overlay system for any TSR application. For general applications, I would let the importance of the TP5 compiler features dictate which overlay system to use.

Finally, in that "best of all possible worlds," programmers everywhere would produce a ground swell of phone calls to TurboPower Software convincing them that a viable market exists for two different overlay systems in a TP5 environment. But then, we can't have everything, can we? ♦

Bill Madison is an independent computer consultant whose clients include Amtrak, the National Institutes of Health, the National Oceanic and Atmospheric Administration, the National Weather Service, the U.S. Department of Health and Human Services, and R.J. Reynolds Tobacco Co. Bill has been a member for nine years, and a chairman for seven years, of ANSI/X3/SPARC, the staff planning committee for the group within ANSI responsible for the development of industry standards relating to information systems. Readers may contact Bill at W.G. Madison and Associates, Ltd., P.O. Box 898, Greenbelt, MD 20770, 301-552-7234 (CompuServe 73240,342).

Software Tools
For Programmers & Non-Programmers

★ NEW ★ VERSION **Opt-Tech Sort**™ ★ NEW ★ VERSION

Opt-Tech Sort is a high performance Sort/Merge/Select utility. It can read, sort and write a file faster than most programs can read the data. Example: 1,000 records of 80 bytes can be read sorted and a new file written in less than 9 seconds (IBM XT). Opt-Tech Sort can be used as a stand-alone program or Called as a subroutine to over 30 programming languages.

Numerous features are included. A partial list of features includes: The ability to process files of any size. Many filetypes are supported including Sequential, Random, delimited, Btrieve, dBASEII & III and many others. Up to 30 key fields can be specified over 16 different types of data are supported. Powerful record selection capability allows you to specify which records are to be included on your output. Record reformatting allows you to change the structure of your output record and to output special fields such as record numbers for use as indexes. **Newest Features: Faster than ever! LIM EMS support. User exits to pass or receive records. Less disk space required and much more!** MS-DOS $149.

On-Line Help™

On-Line Help allows you to easily add Help Windows to all your programs. On-Line Help is actually two help packages in one. You get BOTH Resident (pop-up) and Callable help systems.

The resident version allows you to add help to any system. Your Help system is activated when the "Hot Keys" that you specify are pressed. You can then chain between help windows in any manner you desire. Upon clearing the help windows you are right back into what ever program you left.

The callable version allows you to easily display help windows from your programs. A simple Call to the help system makes the window appear. On-Line Help is callable from over 20 different languages.

You have full control over the help window content, size, color and location. MS-DOS $149. Demo $10.

Scroll & Recall™

Scroll & Recall is a resident screen and keyboard enhancement. It allows you to conveniently scroll back through data that has gone off the top of your display screen. Up to 27 screens of data can be recalled or written to a disk file (great for documenting systems operations). Also allows you to easily recall and edit previously entered DOS commands without retyping.

Scroll & Recall is very easy to use. Its a resident utility that's always there when you need it. MS-DOS $69.

Visa, M/C, AMEX, Check, Money Order, COD or Purchase Orders accepted

To order or to receive additional information just call and receive immediate highly qualified attention!

Opt-Tech Data Processing
P.O. Box 678 / Zephyr Cove, NV 89448
(702) 588-3737

Number 16 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

From Pascal to Modula-2

The TopSpeed Modula-2 Environment

by
John J. Newlin

Dr. Niklaus Wirth, the father of Pascal, formed a project team in 1977 to design a modern computer system. One product of this development was a new programming language, now known as Modula-2. Quite naturally, Modula-2 is a strongly typed language that contains many similarities to Pascal. In creating Modula-2, Dr. Wirth greatly extended the support for structured, modular programming that he had built into Pascal.

Pascal was slow to gain acceptance as a serious programming language until the success of Borland's Turbo Pascal. I was one of the early purchasers of Turbo Pascal. Its compiler technology was so impressive that I asked a technical-support person at Borland if the company had plans to produce a Modula-2 compiler. Borland answered affirmatively, and, as many Turbo buffs know, rumors of a Turbo Modula have abounded for the past several years.

Indeed, such a project was in the works at Borland. In 1987 Niels Jensen, a founder of Borland, and two former Borland programmers formed a spinoff company called Jensen and Partners International (JPI). JPI then purchased the rights to the partially completed Modula-2 compiler and the code-generation technology that the partners had developed while at Borland. That compiler is now a finished product, currently marketed by JPI as TopSpeed Modula-2.

Right out of the box, TopSpeed Modula-2 is a polished, attractive product. The compiler files are delivered on three 5.25" diskettes (3.5" diskettes are available) that include system, library object, and library source. Score one for JPI for delivering the library source at no extra charge! System documentation is comprised of two manuals: a user's manual and a language tutorial.

The JPI Modula-2 package is *not* for novice programmers. Although the documentation is excellent, it is lean and mean, directed more toward programmers familiar with separate compilation and linking. I see the package as ideally suited to producers of serious programs with Borland's Turbo Pascal 4.0/5.0 or Turbo C. Experienced C programmers would probably make the transition to TopSpeed Modula-2 as easily as they would to Turbo Pascal.

The user's manual is divided about equally between explaining operation of the integrated environment and defining the TopSpeed language, compiler, and supporting libraries. Even with the numerous code examples throughout the manual, it's still lean and mean. The indexing is fairly complete, but when I tried to find an explanation and example of aliasing, the index pointed me to the alias compiler directive switch. A thorough search of the manual failed to locate any more than a couple of hints about invoking this important feature (important because of the strict syntax requirements of Modula-2). I called JPI and learned that any defined entity may be aliased by placing a statement in the CONST section of the importing module as follows:

```
CONST              Alloc ::= Storage.Allocate;
```

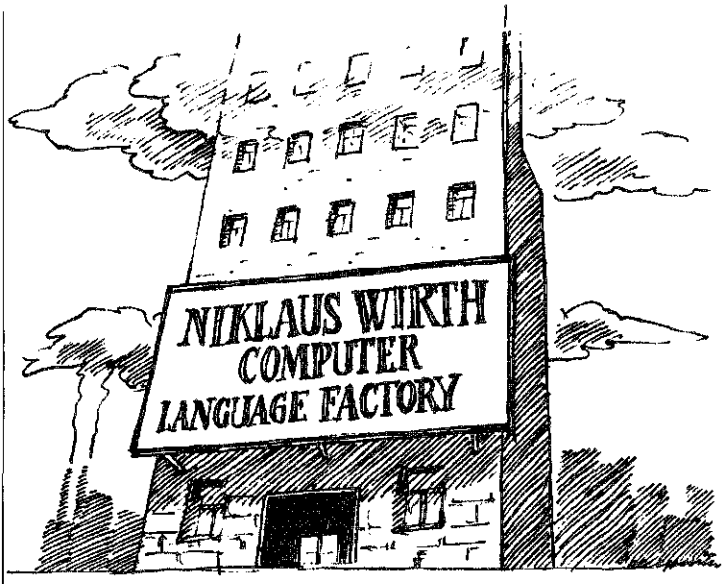
The double-colon–equal sign symbol combination directs the compiler to equate Alloc with a call to Storage.Allocate. Thus, using aliases can save a lot of laborious typing.

Another weakness of the manual is its incomplete listing of TopSpeed's various additional extensions to Modula-2. In some cases, the manual explains that a specific entity is an extension, but most users concerned about portability are going to need a list of the TopSpeed deviations from Wirth's current standard. JPI informs me that it plans to correct this in the next manual upgrade.

The manual's strength lies in its support for users who are not familiar with Modula-2. Chapter 2, titled "Case Studies," presents a number of code examples that begin with the classic "Hello World" program and become progressively more complex. The examples and accompanying explanations are designed to make the user increasingly familiar with the elements of Modula-2 as implemented in the TopSpeed system. This chapter is not a tutorial, however. K. N. King's *Language Tutorial* manual is provided for that purpose. Many compiler packages include third-party tutorials, but these are usually general-language tutorials not specifically related to a particular compiler. In this case, Mr. King's work is pointed specifically at the TopSpeed system. That's a real plus and another score for JPI.

The TopSpeed's system foundation is a multiwindow, integrated programming environment similar to that implemented in Turbo Pascal 4.0/5.0. It's slick, it's fast, and it's functional. Modula-2 programs can be created, compiled, linked, and executed without leaving the environment. In addition, the extensive tree of pop-up windows and menus provides complete knowledge and control of the system. The advantage of Modula-2 programming lies in the design and coding of individual, separately compiled modules. Maximum support for the methodology is provided by the TopSpeed environment. The heart of the integrated environment is the editor—and what a great editor it is! Four separate editing

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



windows are available for editing four different files simultaneously. Switching between the editing windows is accomplished with a single, lightning-fast keystroke. Not only is multifile editing provided, but these are virtual editors that permit accessing a file up to one-half megabyte in size! It's also possible to use the separate editing windows to view four different parts of the same source file. Changes made through one window are automatically inserted in the other three. Now that's source editing power!

But that's not all, by a long shot. You can configure TopSpeed's integrated environment to your liking. If you don't like the placement, the size, or the colors of a particular menu or window, you can change it. All of the editor functions, including commands and key assignments, can be customized readily to suit your tastes. I say readily customized, because TopSpeed has taken a somewhat different approach to configuring the environment than that used in other systems. Instead of a stand-alone installation or configuration program, ASCII definition files are provided that can be modified quickly and easily from the editor.

In my opinion, the only additional feature needed to enhance the editor is a macro-definition capability. For you to understand this capability's importance, I must digress briefly to Modula-2 syntax. Creating Modula-2 source code tends to be a bit clumsy and keystroke intensive. There are two reasons for this. The first is that Modula-2 syntax is case sensitive. A variable named Count is treated as separate from one named COUnt. Additionally, all reserved words must be entered in caps. The second reason is related to Modula-2 qualifiers. If you import an entire module, (for example, IMPORT MyModule;), then you must reference every entity in that module with a qualifier. For example, suppose MyModule contains a procedure named MyProc. The programmer is required to reference this procedure as MyModule.MyProc. That's a lot of typing that the inclusion of a macro facility could alleviate. I'm not advocating complete context-sensitive automation, just the ability to define and invoke string macros. TopSpeed *does* include helpful editor commands that will change a cursor-designated word to all caps as well as change the case of a word to that of the first occurrence of the word in the file. Compiling and linking a program from the editor is fast and easy. You can invoke the compiler either through the pop-up menu system or by pressing Alt-C. Alt-R will compile and run the program in memory, and Alt-L invokes the Linker. JPI's advertised compiler speed of 4000–6000 (8MHz AT) lines per minute is not in the Turbo 5.0 class, but it's fast enough not to cause significant delay in program development. Unlike Turbo 4.0/5.0, no option exists to keep the linker in memory with the compiler. Also unlike Turbo, the linker is a truly "smart" linker. Turbo will import all the code of an assembled module, even though some of the assembly procedures are not referenced. The TopSpeed linker only imports those entities—types, constants, variables, procedures, and so on—that are referenced by the importing module, whether compiled or assembled. JPI provides what I think is a very nice compiler feature. Instead of ceasing compilation at the first error discovered, the TopSpeed compiler continues to the end of the source file or until 50 errors have been discovered and reported. At the end of the compilation, a separate error-editor window is presented to the programmer, with the cursor positioned on the first error. With a single keystroke, the programmer can then immediately move to each subsequent error, correct it, and move on to the next. The standard system does not contain an integrated debugger, but JPI offers a stand-alone debugger, called the Visual Interactive Debugger, or VID. Unfortunately, I haven't had the time to exercise VID, but it appears to be a very powerful development tool. Another very helpful feature is the automatic MAKE facility resident in the compiler. Module dependencies are automatically calculated, and out-of-date modules are recompiled. For example, if a definition file is changed, the automatic MAKE facility will ensure that all modules importing the altered definition will be compiled. A manual MAKE may be invoked from a menu; a complete module-by-module BUILD facility is also available.

JPI not only provides a complete set of library modules with the compiler but provides the source code as well. That's a big plus as far as I'm concerned. The TopSpeed library contains the following 13 modules:

```
SYSTEM          Lib          FIO          ProcTrace
AsmLib          Storage      IO
MATHLIB         Process      Window
Str             Graph        FloatExec
```

Modula-2 programmers will recognize most of these as standard library modules, but one nonstandard module deserves special mention. I know of no other basic compiler package that includes a complete windowing facility. The programmer usually has to use third-party support to obtain high-level window routines or write his own. TopSpeed's Window module is a powerful window-management system that allows the programmer to create, stack, move, change, and write to various windows.

The *tour de force* of Modula-2 is, in my opinion, its intrinsic support for creating multitasking programs. Standard Modula-2 routines such as NEWPROCESS, TRANSFER,

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

and IOTRANSFER reside in the SYSTEM module per language convention. But to accomplish true multitasking (time slicing on a single CPU), the programmer must utilize a scheduler to manage the coroutines. TopSpeed provides just such a scheduler in the Process module. The procedures contained in this module comprise a powerful set of tools for creating and managing coprocesses. Processes can communicate through shared global variables or via signals. Signals are used as a means of process synchronization. JPI provides a rich assortment of compiler directives, giving the programmer full control of the compilation and linking process. For example, there is a directive that controls which 80x86 registers will be preserved during a procedure call. Another directive enables or disables the C language calling convention for procedures to permit the linking of C object code (TopSpeed uses standard .OBJ file format). The programmer can even designate, by name, in which data segment to place the global variables declared in a module and in which code segment to place the code. There is, however, no control over data alignment within records. By default, record field data is byte aligned, and that cannot be changed.

Excellent heap-management tools are provided in this compiler. In addition to the standard Modula-2 ALLOCATE and DEALLOCATE procedures, TopSpeed includes the capability to create and manage separate heaps. The programmer can create new heaps, change their sizes, and deallocate them. Part of this convention permits the assignment of the new heap to a specific base segment. The standard heap-allocation procedures return a segmented address for this purpose. Pascal programmers will appreciate the inclusion of language extensions, such as NEW and DISPOSE, as predefined procedures.

I have neither the space nor the time to touch on all the features that are packed into the compiler. In my opinion, a compiler should be judged on two main criteria: First, is the system user friendly—meaning does the design and implementation of the user interface enhance the programmer's efficiency? I feel that Modula-2 is one of the very best in this test. It doesn't get in your way, yet the system is sophisticated and powerful enough to support production of large, complex projects.

The second criteria is, of course, the quality of the code produced. The following benchmark tables address that issue:

SIEVE BENCHMARK

ITERATIONS	TP4	TP5	JPI
25	6.37	6.4	2.41
50	12.74	2.96	4.88
EXE FILE SIZE	1504	1440	1700

ACKERMAN BENCHMARK

INPUT VALUES	TP4	T5	JPI
(3,6)	2.2	2.26	1.98
(3,7)	9.17	9.17	7.27
EXE FILE SIZE	1456	1392	1691

These figures were derived using an AST Premium/286 running at 8 MHz. The programs were run in memory with range and error checking disabled. In each case, compilation to the disk .EXE file was effected after removing all timing and output code. Don't ask me to explain why the TP5 figures are not much better than those produced by TP4. I double-checked the code and environment and ran the programs 20 times each, with the same results. You can see from the tables that the speed of the code produced by the JPI TopSpeed compiler and linker is superior to that of Turbo 4.0/5.0. I attribute the difference in .EXE file sizes between TopSpeed and Turbo to TopSpeed's automatically importing its SYSTEM module whether it is declared or not. I did not have the time to construct and run any floating point benchmarks, but results of other such tests suggest that TopSpeed retains its superiority.

I was surprised to find that this compiler does not support in-line code. When queried about that, a JPI representative explained that in-line assembly language would compromise the compiler's optimization techniques. Considering the speed of the compiled code and the ability to link separately-assembled modules, in-line code support is really not a problem with this compiler. I feel that the inclusion of such language extensions as absolute variables, string handling, heap and quick sorting, port access, and DOS service access to be much more important. TopSpeed has all of these and more. For $59.95, JPI offers an add-on package they call the "TechKit." The TechKit includes a high-speed assembler and several additional Modula-2 modules. The assembler is designed specifically to support the TopSpeed Modula-2 development environment and in that regard supports import and export of entities to and from compiled modules. Library modules included in the TechKit are TSR, RS, BIOSIO, Int24, and LIM. These module names provide obvious clues to the nature of their functions: TSR provides memory resident support, RS contains asynchronous communications routines, BIOSIO extends keyboard interface support, Int24 is a critical error handler, and LIM offers several basic EMS (expanded memory) procedures. Earlier in this review, I mentioned two primary criteria on which to judge compiler systems. Based upon these criteria, I judge the TopSpeed Modula-2 compiler as an outstanding software development system. The TopSpeed integrated environment is friendly, a pleasure to use, and features the best source code editor one could hope to find. In addition, TopSpeed compiled code is extremely tight and fast. As a software developer who spends a lot of time one-on-one with a compiler, I am seriously considering making TopSpeed my primary development system. I highly recommend it. ♦

John J. Newlin is a retired naval aviator who works for Hughes Aircraft Co. as a systems engineer. John holds degrees in mathematics and computer science and has programmed extensively in assembly, Turbo Pascal, Marshal Pascal, and Modula-2. He is the author of SCOUT, a memory-resident disk/file/directory manager, and ARCMASTER, a "point-and-shoot"-style interface between the user and ARC and de-ARC programs. John is a charter member of the Association of Shareware Professionals. Readers may write to John at the following address: John J. Newlin, 6198 Agee St. #71, San Diego, CA 92122.

Jensen & Partners International, Inc.
1101 San Antonio Road, Suite 301
Mountain View, CA 94043
1-800-543-5202; (in Canada) 1-800-543-8452
TopSpeed Modula-2 Compiler Kit $99.95
TechKit $59.95
Visual Interactive Debugger (VID) $59.95
Complete package $199.95

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

"TOPSPEED EARNS A STANDING OVATION!"

— Kent Porter, Dr. Dobbs Journal

"...TopSpeed is surely one of the finest new products introduced to date in the PC arena...DDJ doesn't give unqualified raves very often, but there's no question about it in this case; JPI's TopSpeed Modula-2 is first-rate."

Kent Porter
Dr. Dobbs Journal

"JPI Modula-2 looks like another classic in the making. It generates code as good as or better than leading C compilers and the programming environment is a genuine pleasure to use."

Dick Pountain
BYTE Magazine

"I liked all of the hard-disk space that was recovered after I deleted my BORLAND, MICROSOFT, and LOGITECH compilers, because with TopSpeed Modula-2 all the rest are obsolete."

Robert D. Randall
Donnelley Marketing

In England and Europe contact:
Jensen & Partners UK Ltd., 63 Clerkenwell Road, London EC1M 5NP. Phone: (01)253-4333. Compiler Kit: £59.95, TechKit £34.95, VID £34.95, DOS 3-Pack £119.95

Handling charges: In UK please phone for VAT and P&P. In Europe, add £6 for up to 3 products, £2 for each add'l product.

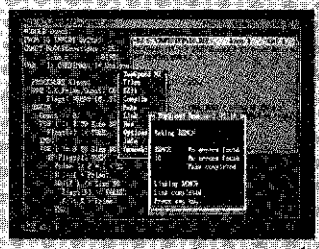
The successor of Pascal: JPI TopSpeed™ Modula-2 produces better code than Microsoft C, Turbo C, Logitech Modula-2 and Turbo Pascal 4.0.

Debug with VID: The easy-to-use, source-level debugger. Single-step and trace through source in multiple modules. Examine and modify all variables in symbolic form, including arrays, records, enumerated types and pointers. Point and shoot breakpoints including "One-shot," "Sticky," "Delayed," and "Monitor." Watch both variables and Modula-2 expressions during execution. Automatic variable trace of all variables accessed, and assembler, registers and procedure call-stack trace windows. Package includes symbolic disassembler and execution profiler. 72-page manual.

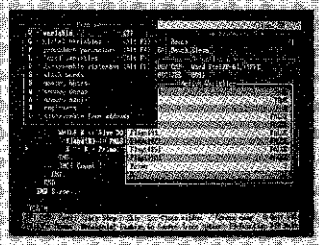
The Compiler Kit includes: High-speed optimizing compiler (3,000-5,000 lines/min. on a PC AT 8MHz), integrated menu-driven environment with multi-window/multi-file editor, automatic *make*, fast smart linker. All Modula-2 sources to libraries included. BONUS: Complete high-speed window management module included with source. 258-page User's Manual and 190-page Language Tutorial.

The TechKit™ includes: Assembler source for start-up code and run-time library, JPI TopSpeed Assembler (30,000 lines/min.), TSR module, communications driver, PROM locator, dynamic overlays, and technical information. 72-page manual.

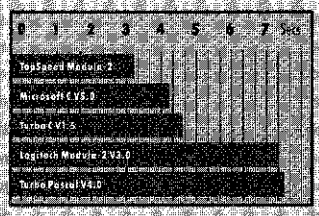
System Requirements: IBM PC or compatible, 384K available RAM, two floppy drives (hard disk recommended).



TopSpeed's seamlessly integrated environment.



VID (Visual Interactive Debugger): power without complexity.



Steve benchmark measured by the British Standards Institution (BSI) — 25 iterations on an 8MHz AT.

Compiler Kit $99.95
TechKit $59.95
VID $59.95

DOS 3-Pack $199.95
(Compiler, TechKit & VID)

To Order:
In the US, call:
1-800-543-5202
In Canada, call:
1-800-543-8452
Or mail us your order with a check, money order, or VISA/MC information. 30-day unconditional money-back guarantee.

Shipping & handling charges:
In North America: add $5, plus $2 for each add'l product. CA residents please add applicable sales tax. Overseas: add $20, plus $8 for each add'l product.

Jensen & Partners International

1101 San Antonio Rd.
Suite 301
Mountain View, CA 94043
Phone: (415)967-3200

TopSpeed and TechKit are trademarks of Jensen & Partners International. Other brand and product names are trademarks or registered trademarks of their respective holders.

Number 249 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Which Code Is Faster?

by
Ethan Winer

No matter what the language, we all strive to create programs that execute in the least amount of time while using the fewest number of bytes. Unfortunately, many programmers who use a high-level language have no way to determine which statements are the most efficient.

The purpose of this article is twofold. We will discuss various BASIC statements to see which ones are faster and smaller. Then we will examine the actual machine instructions generated by QuickBASIC 4 and compare them with the original BASIC source code.

Besides seeing first hand how a compiler operates, we'll also learn about Microsoft's excellent CodeView debugger in the process. CodeView is included with a number of Microsoft languages, including BASCOM 6 and current versions of the macro assembler.

All of the code examples presented here will be shown for stand-alone BASIC programs that do not require BRUN runtime library support. Also, integer variables will be used exclusively. Most programmers use integers whenever possible, and this will prevent the examples from being obscured by floating point math operations.

String Operations

String operations in a QuickBASIC program consist of many calls to routines contained in the BCOMxx.LIB library file. This makes them extremely convoluted and, thus, difficult to trace through. However, manipulating strings efficiently in BASIC is often more a matter of common sense than anything else.

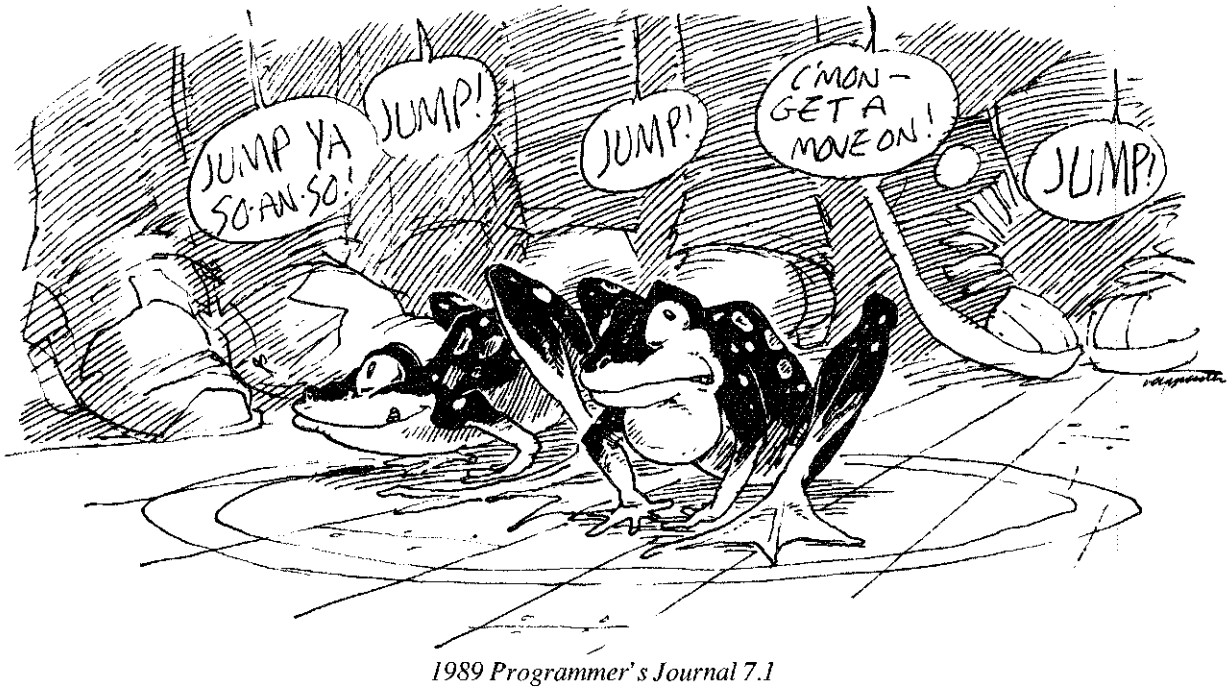
Unlike most other high-level languages, BASIC allows strings to have varying lengths. Whereas languages such as Pascal and C require you to declare all strings and their lengths ahead of time, BASIC is much more forgiving. Of course, nothing comes for free, and so it is with strings in BASIC.

Every time a string is assigned or reassigned, a new memory location is used to hold the contents. Even if the new string is the same length as or shorter than it had been previously, BASIC abandons the memory it occupied and allocates a new space in "near" memory. Near memory is where all nonarray variables are stored; it is limited to 64K.

Because finding a new place to store a string takes time, this is one of the most inefficient operations BASIC does. Therefore, for our first example we'll look at two very different ways to build a string.

Let's say you intend to read a line of characters from the display screen and store them into a string. One method would be to append new characters in a loop like this:

```
S$ = ""                 'clear S$
FOR X = 1 TO 80         'read the top line
    S$ = S$ + CHR$(SCREEN(1, X))    'build the
                                     string
NEXT
```



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Besides being slow because new memory must be found for each new version of S$, this method leaves 3240 bytes of memory abandoned by the preceding assignments. Of course, BASIC will eventually reclaim this memory, but that takes even more time. Because BASIC leaves old string data lying around as it works, it must reorganize its string space from time to time. This process is called *garbage collection.*

Understand that variable-length strings are not a bad thing. Indeed, this is just one of the many advantages BASIC has over less capable languages like C and Pascal.

LSET, RSET, and MID$

BASIC provides two other ways to assign characters to a string—LSET and RSET, and the statement form of MID$. Where the example above must constantly assign a new string for each character that is appended, LSET, RSET, and MID$ let you insert characters into an existing string. Thus, a single string of the correct length may be assigned only once, and the new characters will be placed into subsequent locations.

Using MID$ makes the most sense for a screen-reading routine. It would be coded like this:

```
S$ = SPACE$(80)     'set aside 80 characters
                     in S$
FOR X = 1 TO 80     'read the top line
    MID$(S$, X, 1) = CHR$(SCREEN(1, X))
                    'insert each character
NEXT
```

In a test where 5000 characters were inserted, MID$ filled the string in only 1.2 seconds, as opposed to 7 seconds for the first example.

LSET and RSET also replace characters in an existing string; however, they also clear all characters that are not being replaced:

```
X$ = "Testing 1, 2, 3"
LSET X$ = "Hi"        'now X$ = "Hi          "
RSET X$ = "Lo"        'now X$ = "         Lo"
```

As you can see, LSET left-justifies the string being inserted, and RSET right-justifies it.

Another important way to speed up string operations is to use their ASCII values whenever possible. For example, suppose you need to search through a string array for a particular item. One common method would be:

```
FOR X = 1 TO ArraySize
    IF Array$(X) = Search$ THEN
       PRINT "Found at element"; X
       EXIT FOR
    END IF
NEXT
```

A better approach would be to determine the ASCII value of the string being searched for and compare that to the value in the array:

```
A = ASC(Search$)
FOR X = 1 TO ArraySize
    IF ASC(Array$(X)) = A THEN
       IF Array$(X) = Search$ THEN
          PRINT "Found at element"; X
          EXIT FOR
       END IF
    END IF
NEXT
```

Though this takes a bit more effort to code, the resulting improvement can be substantial. Because BASIC operates on integers much more quickly than it can on strings, a lot of unnecessary testing at the string level will be avoided.

There are certainly other, similar, ways in which you can increase the string performance of a program. As I said earlier, many solutions present themselves once you understand how BASIC operates. On that note, let's take a look inside QuickBASIC to see some of the ways in which your source code is translated into executable machine instructions.

Variable Assignments

Imagine that you are writing a program and that at some point several variables must be assigned to the same value. One way would be to assign the value repeatedly, as shown below:

```
A = 13
B = 13
C = 13
D = 13
E = 13
```

Another possibility is to assign the first variable from a constant and the remaining variables from the first one:

```
A = 13
B = A
C = A
D = A
E = A
```

The second example is much more efficient, but let's see why.

When QuickBASIC compiles your source program, it assigns and remembers an address for each variable it encounters. This lets it replace references to those variables with addresses that the PC's processor can understand. Consider the following assignment statement:

```
A = 13
```

This is compiled to the following machine and assembler instructions:

```
C70636000D00        MOV WORD PTR [0036], 13
```

As you can see, QuickBASIC has placed the variable A at the absolute address 36, and a six-byte assembler instruction is

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Six Five easy ways to boost your BASIC

PROBAS™ updated

Professional Basic Programming Library

Announcing PROBAS Version 3.0, now with over 335 assembly routines to really kick QuickBASIC and BASCOM into high gear. BYTE magazine calls PROBAS a "Supercharger for QuickBASIC". Thousands of programmers rely on PROBAS to make their life easier and to enhance their programs with features like:

- **An 800-page 3-part manual**
- **Full-featured windowing**
- **Screen snapshots (Text & Graphics)**
- **String, array, and pointer sorts**
- **Lightning-fast file I/O**
- **Full mouse support**

Create dazzling screens in text mode, CGA, EGA, VGA or Hercules graphic modes. Save and restore screen snapshots to arrays, EMS memory or files. Full featured windowing to meet the most demanding jobs. The PROBAS system of virtual screens allows you to draw full or partial screens to memory, and then snap them on in an eyeblink. You can even create vitural screens far larger than the display screen.

Sick of running out of string space? Store hundreds of K in numeric arrays or megabytes in extended or expanded memory. Tired of using a kludgy SHELL to DIR to read a directory or archive files? Scan subdirectories or ARC files using wild-cards and store thousands of file names, dates, and times. Wish you could drag a window containing text or a menu around the screen with a mouse? It's easy!

PROBAS gives you a complete set of blazingly-fast file routines. Read or write huge chunks of data at a clip, with file locking and error handling so that you can even use them in subprograms. You'll never want to use BASIC's file I/O again! Sort data with lightning fast array and pointer sorts. Search files or arrays at assembly speeds. PROBAS also has over 200 other essential services including handy string, date, time, directory and array manipulation, string, screen and data compression, full mouse support, valuable equipment and input routines and faster replacements for most BASIC commands.

Whether you are a professional or a novice, PROBAS will boost your BASIC in ways you never dreamt possible. PROBAS allows professionals to save time and work and lets novices write professional-quality programs quickly and easily. After all, how much is a few hundred hours of your time really worth?

For all versions of QuickBASIC and BASCOM including BASCOM 6.0 for OS/2. ***Just $135.00!***

PROREF™

On-Line Help For PROBAS

PROREF provides pop-up help for the routines in PROBAS and is an extension of the QuickBASIC programming environment. Find help on any routine with a few keystrokes or mouse clicks. Pop-up an ASCII chart, calculator, scan code module, box diagram, your own help information or almost any DOS program via a hot-key. ***Just $50.00!***

PROSCREEN™

Professional Screen Management System

PROSCREEN is a full-featured screen generator/editor that will save you more design and coding time than you ever thought possible. PROSCREEN treats screens like a word processor treats text to provide complete control over characters, colors, and placement. Design input screens with up to 130 fields and 19 pre-defined and 2 user-defined masks. Use PROBAS or the included BASIC/Assembler subroutines to access the screens. No kludgy code generators here! Comes with subroutine source, extensive on-line help, and a 285 page manual. ***Just $99.00!***

new! PROMATH™

PROMATH is a collection of over 150 high-level routines that provide mathematical functions and operations for programmers who often work in mathematics, science, or engineering. Complex variables, real and complex matrices, real and complex trigonometric and hyperbolic functions and their inverses, solution of linear equations, integration, differential equations, Fast Fourier transforms and many other useful routines are provided.

For years Fortran has been the language of choice for scientific and engineering applications, but it lacks many of the useful features of QuickBASIC. PROMATH contains most of the Fortran mathematical and numeric functions and allows you to easily translate Fortran code to BASIC or write new programs in BASIC while retaining Fortran's numerical prowess.

The PROMATH manual is over 200 pages and provides a complete description of each routine, including any algorithm and the mathematical formula the routine uses, shown in standard notation. For QuickBASIC 4 and BASCOM 6 only. ***Just $99.00!***

PROBAS™ TOOLKIT

The TOOLKIT is a collection of assembly and BASIC modules that use the PROBAS library to save you even more hours of grunt work. Why spend hundreds of hours reinventing the wheel when you can just plug in TOOLKIT modules like:

- **Menu Generators**
- **Fast B-tree indexing**
- **Mini-editor with word-wrap**
- **Patch .EXE files**
- **Protected storage areas**
- **Julian date routines**

The TOOLKIT also includes clock, calendar, windowing, BCD math routines and much more, complete with BASIC source code and a comprehensive manual. The PROBAS TOOLKIT adds capabilities and helps conserve your most valuable asset of all—time! Requires PROBAS . ***Just $99.00!***

PROBAS™ TELECOMM TOOLKIT

The PROBAS TELECOMM TOOLKIT is a collection of high-level communications modules that you plug into your code to provide popular file transfer protocols, terminal emulations, login scripts and baud rates up to 115,200 baud. You get:

- **Xmodem/Modem7/Xmodem-1k**
- **Ymodem (single and batch)**
- **CRC-16 and Checksum**
- **VT52, VT100, ANSI BBS etc.**
- **Auto Dialer & data base**
- **Documented BASIC source**

Why use clumsy SHELLs to complex terminal programs when you can plug just the communications routines you need into your code? Implement just the features and commands you want. Requires PROBAS. ***Just $75.00!***

Our thirty day, money-back guarantee assures you the highest quality and our technical support staff is always ready to help. Try our BBS at **(301) 953-7738** or give us a call at:

HAMMERLY
COMPUTER SERVICES, INC.
8008 SANDY SPRING ROAD • LAUREL, MD 20707
(800) 343-7484

Add $5.00 per item ($8.00 Canada) for shipping per order. Europe: $39.00 for 1st item plus $5.00 for each additional item. Visa, M/C, C.O.D. (US Only) checks and approved POs accepted. Trademarks PROBAS, PROREF, PROSCREEN, PROMATH, Hammerly Computer Services, Inc. QuickBASIC, BASCOM: Microsoft Corp.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

required to assign it. (The bytes of machine language that QuickBASIC generates are shown to the left of the equivalent assembler mnemonics.) Two bytes are used for the actual instruction, another two to specify the address (0036), and two more for the value 13.

Thus, QuickBASIC will code the first example like this:

```
A = 13     C70636000D00    MOV WORD PTR [0036], 13
B = 13     C70638000D00    MOV WORD PTR [0038], 13
C = 13     C7063A000D00    MOV WORD PTR [003A], 13
D = 13     C7063C000D00    MOV WORD PTR [003C], 13
E = 13     C7063E000D00    MOV WORD PTR [003E], 13
```

Notice that two-byte memory addresses and values are always shown with the lower byte first; that is, the address 0036 is represented in memory as 3600, and the value 0013 is shown as 1300. This is not inconsistent when you think about it, because the lower byte is always stored in the lower memory location. Also notice that QuickBASIC is very organized as it assigns variable addresses—and each variable address occupies the next available memory location.

The second example is compiled to the following machine code, and it is immediately obvious that fewer bytes are needed. All but the first of these instructions use the value held in a register (AX), which is much faster than using a constant value contained within the assembler code instructions.

```
A = 13     C70636000D00    MOV WORD PTR [0036], 13
B = A      A13600          MOV AX, WORD PTR [0036]
           A33800          MOV WORD PTR [0038],AX
C = A      A33A00          MOV WORD PTR [003A],AX
D = A      A33C00          MOV WORD PTR [003C],AX
E = A      A33E00          MOV WORD PTR [003E],AX
```

Because QuickBASIC is very smart, it creates code to retrieve the value 13 from A only once and then uses that value to assign all of the remaining variables. The first example comprises 30 bytes of code, but the second needs only 21.

QuickBASIC lends itself to another clever technique when incrementing and decrementing variables. The simplest type of compiler would take a statement such as X = X + 1 and create code something like this:

```
MOV  AX,[0036] ;move the value of X (address
                36) into AX
ADD  AX,1      ;add 1 to AX
MOV  [0036],AX ;move AX back into X
```

But QuickBASIC is much smarter than that, generating code to directly increment the variable X:

```
INC  [0036]    ;increment X
```

Both INC (increment) and DEC (decrement) are instructions the PC's processor can execute directly, and they are much faster than the ADD or SUB commands.

Array Assignments

Array variables are often initialized in a FOR/NEXT loop, so let's look at that next. The examples below show how QuickBASIC codes a simple FOR/NEXT loop, as well as how variables within it are assigned. In this example, we show the translation much as CodeView does; that is, each line of BASIC source code is shown, followed by the compiled machine instructions.

Notice that the DIM command does not generate assembler code, because memory for a Static array is set aside when the program is compiled.

```
DIM Array(10)
FOR X = 1 TO 10
    B80100          MOV AX,0001
    E90B00          JMP Label2
Array(X) = 0
                Label1:
    8BF0            MOV SI,AX
    D1E6            SHL SI,1
    C78436000000    MOV [SI+0036],0000
NEXT
    40              INC AX
                Label2:
    A30001          MOV [0100],AX  (100 is
                                    X's address)
    3D6400          CMP AX,0A      (10 is 0A
                                    Hex)
    7EED            JLE Label1
```

In the example above, every element in the array is being assigned a value of 0. QuickBASIC uses the AX register as a loop counter and begins by initializing the AX register to the starting count of 1. This is immediately followed by a jump into the NEXT portion of the code, where the value in AX is tested to see if the ending value of 10 (0A Hex) has been reached. If AX is less than or equal to 10, a Jump is made back into the body of the FOR/NEXT loop.

To determine the address of the current array element being assigned, the program moves the X count value held in AX into SI and then multiplies SI times 2 with the SHL (Shift Left) instruction. Because each integer array element comprises two bytes, any element can be found by adding twice the current count to the address of Array(0). For example, the memory location for Array(10) is 20 bytes beyond Array(0).

The actual assignment is made by adding the computed offset held in SI to the constant 0036, which is the address of Array(0). As you can see, QuickBASIC uses the less-efficient method of assigning each element from a constant value. It would have been much more efficient to first clear another register (perhaps BX) to 0 and then use the register for all of the assignments that follow.

Simple Integer Math

Of course, we shouldn't expect QuickBASIC to be as efficient as a human hand coding in assembly language. The code created by any compiler can always be improved. However, as the short program below illustrates, when it comes to simple integer math, QuickBASIC is extremely efficient:

```
X = 4
    C70636000400    MOV  WORD PTR [0036],4
X = X * X
    A13600          MOV  AX,[0036]
    F7E8            IMUL AX
    A33600          MOV  [0036],AX
X = X * 2
    D1E0            SHL  AX,1
    A33600          MOV  [0036],AX
X = X * 3
    B90300          MOV  CX,3
    F7E9            IMUL CX
    A33600          MOV  [0036],AX
X = X * 4
    D1E0            SHL  AX,1
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
    D1E0            SHL  AX,1
    A33600          MOV  [0036],AX
X = X + X
    01063600        ADD  [0036],AX
```

First, X (which is at address 0036) is assigned from the constant value 4. Then AX is loaded from X, where it stays throughout the remaining assignments. To multiply X * X, QuickBASIC generates an integer-multiply instruction (IMUL) and then places the result back into X. Since QuickBASIC is smart enough to realize that AX still holds the value of X, the remaining operations are all performed on AX.

To multiply X times 2, a SHL (Shift Left) instruction is used. It is much more efficient than a conventional multiply: Whenever the bits in an integer variable are shifted left one position, the value is multiplied by 2; likewise, when the bits are shifted right, the value is divided by 2.

Multiplying times 3 cannot be done by shifting, so the CX register is first loaded with the value 3, and IMUL is then used to perform the multiplication. Once this is done, AX is moved back into X. This continues through multiplying times 4, which again is done by shifting.

Finally, the X + X assignment is handled elegantly by the simple addition of AX to the variable's address. Again, this code is extremely efficient, because QuickBASIC is smart enough to realize that AX still holds the value of X from the previous computations.

Another important optimization that QuickBASIC performs is calculating identical expressions only once. In the example below, A is multiplied times B only once for each pass through the FOR/NEXT loop:

```
A = 9
B = 10
FOR X = 1 TO 10
    Array1(X) = A * B
    Array2(X) = A * B
NEXT
```

When examined under CodeView, the program looks like this:

```
A = 9
    C70662000900    MOV  [0062],9
B = 10
    C70664006400    MOV  [0064],0A
FOR X = 1 TO 10
    B80100          MOV  AX,1
    E91900          JMP  Label2
Array1(X) = A * B
                Label1:
    A16400          MOV  AX,[0064]   (64 is B's
                                      address)
    F72E6200        IMUL [0062]      (62 is A's
                                      address)
    8B366600        MOV  SI,[0066]   (66 is X's
                                      address)
    D1E6            SHL  SI,1
    89843600        MOV  [SI+0036],AX
                                (Array1(X) = AX)
Array2(X) = A * B
    89844C00        MOV  [SI+004C],AX
                                (Array2(X) = AX)
NEXT
    A16600          MOV  AX,[0066]   (66 is X's
                                      address)
    40              INC  AX          (AX = AX +
                                      1)
                Label2:
    A36600          MOV [0066],AX    (X = AX)
    3D0A00          CMP  AX,0A       (10 is 0A
                                      Hex)
    7EDF            JLE  Label1 (jump if less
                                or equal)
```

Because neither A nor B is changed inside the loop, QuickBASIC *should* calculate the result only once ahead of time. Even though many C compilers perform that level of optimization, QuickBASIC is not that smart, so you could easily do this yourself to make the program run faster:

```
A = 9
B = 10
Temp = A * B
FOR X = 1 TO 10
    Array1(X) = Temp
    Array2(X) = Temp
NEXT
```

Using CodeView

All of the examples shown thus far would have been impossible to present without the invaluable aid of Microsoft's CodeView debugger. If you are familiar with the QuickBASIC 4 editing environment, you should have little trouble understanding and using CodeView.

Like QuickBASIC 4, CodeView features break points and watch variables, and it lets you single-step through a program. Also like QB4, CodeView can step over or into subprograms and system interrupts. Understand that CodeView is the *only* reasonable way to debug a complex assembler routine that has gone berserk. While DEBUG can be helpful for simple assembly language COM programs, it is all but useless for debugging an assembler routine that has been added to a QuickBASIC program.

Preparing a BASIC or assembler program for debugging with CodeView is very easy—simply add the /ZI switch to the BC or MASM command:

```
BC basicprogram /ZI
```

or

```
MASM asmprogram /ZI
```

Once the program has been compiled or assembled, it must also be linked. This is done by adding the /CO switch when LINK is started:

```
LINK basicprogram [asmprogram] /CO
```

Though the logic escapes me, the LINK switch for a CodeView program is not the same as for BC and MASM. Indeed, /CO is much more sensible, and it's anybody's guess as to what /ZI is supposed to stand for.

For the most part, CodeView is not necessary if your program is written purely in BASIC. After all, the whole point of a high-level language is to avoid having to deal with registers and instructions to the PC's processor. However, if you are creating assembly language routines meant to be called from

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

QuickBASIC, it is invaluable. Of course, this is also the only reasonable way to see how QuickBASIC works.

Using CodeView with QuickBASIC

To examine the code that QuickBASIC creates, let's walk through one of the sample programs we looked at earlier. Because we already listed the assembler code created by QuickBASIC, we'll just go over the necessary steps here. We'll get to using CodeView on a mixed program of QuickBASIC and assembler routines in a moment.

First, create a BASIC source file, preferably using one of the examples above, or a short program with just a few simple instructions. When you save it to disk, be sure to use the "Save As" option and select "Text." CodeView has no way to decipher the special "fast load" format that QuickBASIC uses as the default save method.

Next, compile the program from the DOS command line using the BC compiler and the /ZI and /O options:

```
BC basicprogram /ZI /O;
```

Once the program has been compiled, it must be linked:

```
LINK basicprogram /CO;
```

Finally, start CodeView like this:

```
CV basicprogram
```

Once CodeView begins, you see a screen similar to the one used by QuickBASIC. CodeView starts up in the "immediate" mode, so you should press F6 once. At this point, you are viewing the original BASIC source instructions for your program. Pressing the F3 key lets you view the equivalent assembler instructions, but don't do that yet!

Every BASIC program begins with a large section of startup code, which we want to skip over first; so press the F10 key to step to the first line of the BASIC source code. When CodeView is showing your BASIC source file, F10 steps to the next BASIC instruction.

Now you can press F3 to view the assembly language statements intermixed with the BASIC source. Each time you press F10 while in the assembly view mode, a single assembler instruction will be executed. Debugging a mix of BASIC and assembler source is only slightly more complicated, as we are about to see.

Using CodeView with Mixed Languages

While using CodeView with a BASIC program can certainly be an enlightening experience, CodeView really shines when you need to follow a call from BASIC to an assembler routine and back again. As with the BASIC examples above, the assembler source must be assembled with the /ZI switch, and you must invoke LINK with the /CO option:

```
BC basicprogram /ZI /O;
MASM asmprogram /ZI;
LINK basicprogram asmprogram /CO;
```

When CodeView is started, the BASIC source will be presented first, and again you should use F10 to step to the first line in the BASIC program. When you get to a call to an assembly (or other) language routine, press F8 rather than

PROTECT INTELLIGENCE WITH INTELLIGENCE

PROTECH KEYs:

- ❑ are totally transparent to the end user
- ❑ allow unlimited back up copies
- ❑ free up disk drives
- ❑ do not interfere with peripheral operations
- ❑ are easy to install
- ❑ protect software developers' revenues
- ❑ provide the highest level of protection available

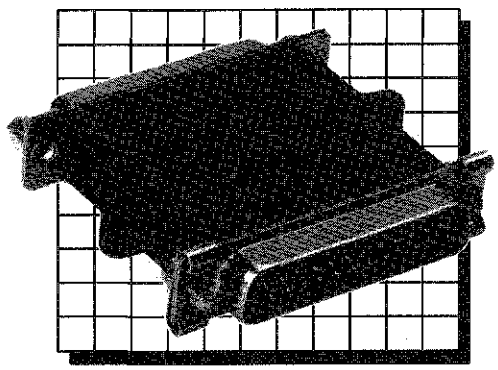
PROTECH's KEYs provide the highest level of protection because of:

- ❑ assembler based routines (not drivers)
- ❑ encrypted routines
- ❑ physically unique hardware KEYs
- ❑ sophisticated software installation advice

THE KEY

The KEY provides effective software protection while insuring customer satisfaction. The KEY is a random response device which is designed for identically reproduced software packages.

SOFTWARE PROTECTION KEYs BY PROTECH



WE HAVE THE *KEY* TO SOFTWARE COPY PROTECTION

For a demonstration package or additional information, please write or call:

1-800-843-0413

THE MEMORY KEY

The MEMORY KEY is a programmable software protection device. Each byte of memory can be addressed in groups or individually. Possible applications for the MEMORY KEY include:

- ❑ modular package control
- ❑ serialization
- ❑ customization
- ❑ access control to PCs

Information stored in the MEMORY KEY can also be altered by your application software in the field. This dynamic writing to the MEMORY KEY opens up unlimited possibilities. Possible applications for the dynamic read/write function include:

- ❑ demo control
- ❑ software leasing
- ❑ updating modules in the field
- ❑ any "counter" operation

PROTECH MARKETING, INC.
1804 W. Southern Parkway • Building A-112
Durham, North Carolina 27707
(919) 490-4970 FAX (919) 490-4974

Number 226 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

F10. As in the QuickBASIC editor, F8 steps into a function or subroutine call, while F10 considers the call to be a single instruction and steps over it.

If the assembler source code is not in the current directory, CodeView will ask you to enter the directory name. In order to show both the source code and the machine instructions at the same time, CodeView must be able to load the original source files.

Watching assembler variables is similar to the same operation in QuickBASIC. Pressing Alt-W brings up the Watch menu, and you simply enter the variable's name. Any time the value of a watch variable changes, it will be reflected in the watch window at the top of the screen.

And that's really all there is to it! As long as you remember to use the /ZI and /CO switches, you will have no trouble tracing through either BASIC or assembly language source code.

. . . Well, that's *almost* all there is to it. There are a few bugs in CodeView you should be aware of. (There are several versions of CodeView in circulation; the bugs in the version I have are mentioned below, just so you'll know they aren't your fault if you encounter them.)

The first problem is that CodeView will not work if SideKick is loaded. Mind you, your program won't crash, but CodeView reports the following error:

```
run-time error R6002 - floating point not loaded
```

This error is not mentioned anywhere in the CodeView manual, and it is in fact a C language error message!

The second bug is probably QuickBASIC's fault, because it affects a TYPE variable that uses long integers. In a test program I was writing, the following TYPE declaration was placed early in the source code:

```
TYPE VLong
     Lo AS LONG
     Hi AS LONG
END TYPE
```

CodeView crashed on this one so badly that the power had to be turned off. However, the solution was simple:

```
TYPE VLong
     Lo1 AS INTEGER
     Lo2 AS INTEGER
     Hi1 AS INTEGER
     Hi2 AS INTEGER
END TYPE
```

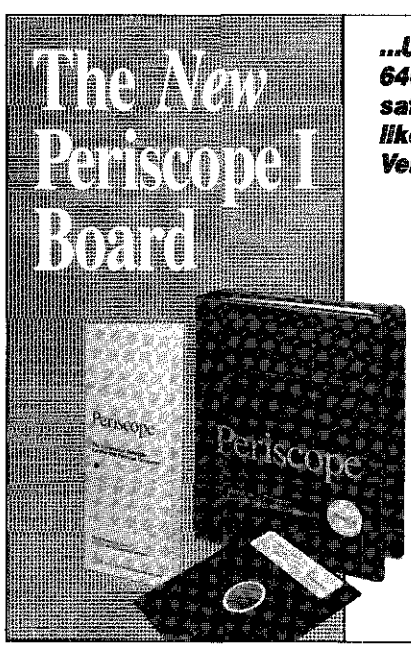
Since I could use any block of data that was eight bytes, having to split each LONG into two integers wasn't a problem. Depending on what you're trying to do, a different workaround may be needed.

Learning More

We have looked at only a few samples of QuickBASIC code that have been compiled to machine instructions. Of course, many BASIC statements are difficult to follow once they have been compiled because of the complexity of QuickBASIC's system of library calls. However, tracing through some of the simpler functions can be very enlightening.

I encourage you to experiment on your own, perhaps looking at such functions as LEN and ASC, to see how these functions work. Indeed, my own curiosity led me to develop more-efficient replacements for these commands. ♦

Ethan Winer is the president of Crescent Software and the author of QuickPak and QuickPak Professional. Ethan can be reached at Crescent Software, 11 Grandview Avenue, Stamford, CT 06905, (203)-846-2500, on MCI Mail as EWINER, and on CompuServe at account number 72657,3070.



...Uses ZERO memory in the lower 640K. Yet it has plenty of room to safely store all debugging information, like symbols, as well as the powerful Version 4 software.

New in Periscope Version 4:

- View local symbols from Microsoft C
- Debug Microsoft Windows applications
- Set breakpoints in PLINK overlays
- Improved source-level support
- Monitor variables in a Watch window
- 80386 debug register support
- Debug using a dumb terminal
- PS/2 watchdog timer support
- Use mixed-case symbols

"The best keeps getting better... Once again, Periscope has raised the industry standard for debuggers!"

—David Nanian
BRIEF co-author

Periscope I includes a NEW full-length board with 512K of write-protected RAM; break-out switch; software and manual for $795.

Periscope II includes break-out switch; software and manual for $175.

Periscope II-X includes software and manual (no hardware) for $145.

Periscope III includes a full-length board with 64K of write-protected RAM, hardware breakpoints and real-time trace buffer; break-out switch; software and manual. Periscope III for machines running up to 10 MHz is $1395. Plus the new Model 1 board, $1995.

Prices subject to change without notice.

Call Toll-Free for free information or to order your Periscope today!

MAJOR CREDIT CARDS ACCEPTED. **800-722-7006**

The Periscope Company, Inc.
1197 PEACHTREE ST.
PLAZA LEVEL
ATLANTA, GA 30361
404/875-8080

Number 6 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Programmer's Tools

from The Software Engineering Store & The Software Family

C

Borland Turbo C v2 ..... New! $99
Borland Turbo C v2 Prof. .....New! $169
Blaise C Tools Plus/5.0 .....$99
Creative Vitamin C $169
Essential Graphics v2 ..... $232
Gimpel PC Lint $99
Ithaca HOOPS $455
Object Oriented Graphics Database
Microsoft C v5.1 .....$299
Microsoft Quick C $68
Novell Btrieve $199
Softfocus Btree/ISAM $99
comprehesive file/index manager with all source
Wordcraft C Workshop $69
C training manual & software
Zortech C++ $99
Optimizing C++ Compiler

Turbo Pascal

Borland Turbo Pascal v5 ..... New! $99
Borland Turbo Pascal Prof. ... New! $169
Turbo Power Professional v5 .. New! $99

Other Languages & Tools

Digitalk Smalltalk/286 ..... New! $164
Borland Assembler & Debug New! $99
Matrix Layout.....$129
Microsoft Macro Assembler.....$105
Microsoft Windows SDK v2.1 .....$328
Mortice Kern MKS Toolkit v2.3 ..... $145
KORN shell & 130 UNIX™ -like utilities
Periscope I 512K v4 .....New! $640
Periscope III v4 .....$1125
Qualitas 386-Max $69
80386 memory manager
Solution Systems Brief ..... $call
Whitewater ACTOR $449
High productivity development for MS Windows

Free Catalog & Newsletter

We carry hundreds of products from dozens of publishers. Call or write for your free subscription to our catalog and newsletter.

Returns

We want you to pay for only the tools you really use. If you order anything featured in this ad (or most of the products in our catalog) you can open and use the product for up to 30 days and still return it if it doesn't meet your needs.

Prices subject to change without notice - ©1988 TSF All Rights Reserved

Multi-Tasking

DESQview *by Quarterdeck* is the only multi-tasking environment that provides complete compatibility for the entire PC hardware family while fully exploiting the the features of the 80286 and 80386. **$99**
QEMM **$55**
API Reference **$54**
API Ref & C Library **$179**

Disk Duplication

Duplicator Toolkit *by Copy Technologies* simplifies and speeds diskette duplication. One step operation to format, copy & verify diskette and print label — faster than just copying with DOS. **$79**

The Software Engineering Store

1-800-443-7176

In Calif: 1-800-635-0969

(415)957-0111

Visa • MasterCard
COD • Check

- Most orders shipped within 24 hours
- Free shipping on orders over $100

575 Mission Street, #600-PJ
San Francisco, CA 94105

The Software Family has a new name!

We've changed our name, but not our ownership, staff or outstanding service. The Software Engineering Store continues our tradition of fair pricing, expert technical support and honest business practices.

386 Hardware

Hauppauge 386XT replaces your XT motherboard to provide top 80386 performance with maximum compatibility. 16Mhz CPU, 1MB zero wait state memory. Provides 99.5% compatibility with both XT and AT hardware and software, and backed by a 30 day money-back compatibility guarantee. Made in U.S.A. One year warranty. **$1495**

Debugging

New version! **Periscope** ***v4*** *by Periscope*: The Periscope debuggers provide the combinations of hardware and software you need to solve really tough programming problems. Periscope includes all the debugging features you need, but with minimum overhead and maximum flexibility. With Periscope you can debug programs which need all memory, have real-time requirements and/or are picky about vector usage.

Periscope I w/512K **$640**
0k memory overhead!
Periscope III **$1125**
hardware breakpoints and trace-back for real-time applications

Technical Support

The Software Engineering Store carefully evaluates features, quality and support before offering a product to our customers. All of the products featured in this ad are backed by excellent technical support from the publisher — in many cases directly from the program developers. All are published by companies with reasonable update policies and with a track record for responding to customer requests for both bug fixes and new features. We further back the products with our own technical support. We are willing and able to intercede with the publisher if your needs aren't satisfied or to research problems caused by the interaction of products we recommend. We can also provide quotes for custom modification of source-licensed products, in-depth technical consultation or custom-developed products.

Number 93 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Proposed ANSI C Language Standard

by
Rex Jaeschke

© *Rex Jaeschke, 1988. All rights reserved.*

Lattice V3.3 for DOS and OS/2

At one time, Lattice was one of the major players in the DOS-based C compiler business. However, in recent years they have seen some stiff competition. They were acquired by SAS Institute, and they have had some financial and other reorganizations, all of which have affected their presence in the marketplace. In fact, when SAS acquired Lattice, Lattice lost their voting status on the ANSI standards committee, since each company is permitted only one voting member. However, that does not prohibit such companies from attending meetings and actively participating in the discussions.

Lattice has not been beating their drum and claiming they are tracking the ANSI standard—although their packaging and documentation might lead you to believe they are. I guess that's just marketing license. In any case, the preprocessor, compiler and library all come up considerably short with regard to ANSI-standard conformance.

The Compiler Options

A few compiler switches are worth mentioning: -cs causes like string literals to be "pooled," with only one copy actually stored; -cu makes plain chars unsigned instead of signed; -cw disables warnings for nonvoid functions having no return values. Be careful about using this: While it may save lots of messages if your code was written before void functions were permitted, it will not tell you if you have made a legitimate error. I strongly suggest that you declare void functions as such, rather than defaulting to ints without returning a value.

The -F option generates function prototypes from existing source; however, this option has shortcomings and bugs. For example, an int function defined with no arguments has an ANSI prototype of int f(void);. The Lattice compiler does *not* put the void inside the prototype, thereby rendering the generated prototype almost useless. Having nothing inside the parentheses tells the compiler not to check the calling argument list when, in fact, it should make sure there are no arguments.

Additionally, the new style of defining a function is not supported. In the old K&R style, any argument having a narrow type was really passed in its widened form; however, when a char argument was declared, it was retained as a char in the prototype generated, while it should have been made an int. Strangely enough, a short argument was correctly made an int, and a float was correctly made a double.

Another bug occurs when an argument is declared as type short *. This was incorrectly changed to int * in the prototype, which could have disastrous affects if you moved your code and prototypes to a machine on which short and int had different representations. A final problem is that each prototype in the output file is terminated with a line feed rather than with a carriage return–line feed combination; the file is not particularly readable when you type it out using the DOS TYPE command. Depending on your editor, you may also have trouble editing it. This option definitely needs a lot of work.

The -x option lets you write all occurrences of external data declarations without the extern keyword. The K&R and ANSI models require one—and only one occurrence—not to have the extern keyword; all other references must have it.

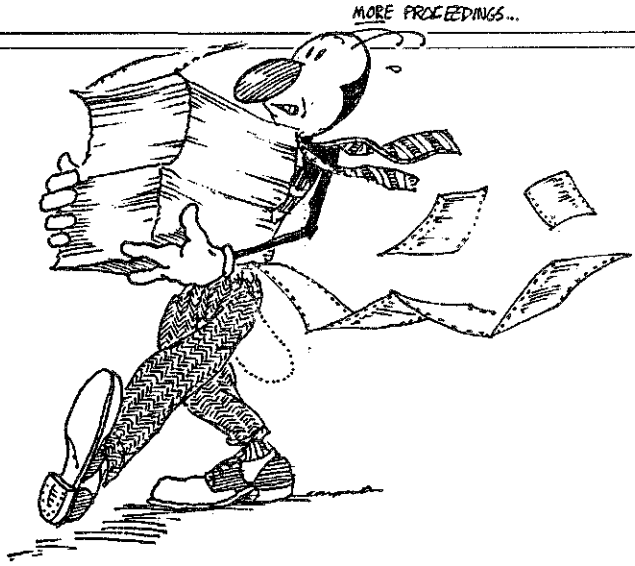
The Preprocessor

The following ANSI C attributes are not supported or had some limitation:

- Macros are limited to 16 arguments.
- Macro arguments are replaced inside strings and character constants.
- A macro can be redefined. The preprocessor stacks the different definitions and produces a warning.
- No #, ##, or "defined" operators are supported.
- No predefined macros exist for __DATE__ and __TIME__. Therefore, these and other macros (__FILE__ and __LINE__) are allowed to be #undefed.
- No support is provided for arbitrary token continuation using a backslash–new line pair.
- Excess tokens on directive lines are ignored.
- Tokens are built incorrectly in certain circumstances.
- Comments are permitted to span source files.
- Explicit, long int constants in #if expressions are a problem —#if 10/0 hung the compiler (and the PC).
- No #elif is supported.
- No #include identifier is supported.
- No #error or #pragma is supported.
- ANSI C permits "sizeof" to be rejected by the preprocessor but does not require it to be. Lattice rejects it.

I encountered a bug when a macro M was defined with arguments and a declaration of int (M)(); was used. The declaration was incorrectly recognized as a call to macro M; however, since the token immediately following the M is not

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



a left parenthesis, it is not a macro call. (I have found this error in numerous compilers.)

The C Language

The following ANSI features are not supported:

- Bit-fields in unions
- Initialization of auto arrays and structures
- The new keywords const, volatile, and signed
- Concatenation of adjacent string literals
- Long double type or long double constants
- Float constants
- Unsigned constants
- Trigraphs
- String literals up to 509 characters

Void pointers are supported.

Function prototype support has been partially implemented. A function declaration is permitted to have type declarations in place of each argument expected. The trailing ellipses are also supported for variable argument lists. Note, though, that the ellipses can also be omitted, leaving a comma followed by a right parenthesis. This is not permitted by ANSI C and was an interim stage of the design during ANSI deliberations. Don't use it. The only real drawback is that dummy identifier names are not permitted in prototypes. Also, the new prototype style of defining functions is not supported.

In numerous examples in the manual, main is declared as a void function. Not only is this never true, but it teaches bad programming practices. If you never return a value from main and you want to be rid of the warning the compiler generates, return a value of zero instead.

The Runtime Library

The library and headers are pretty much K&R vintage, with the usual UNIX stuff thrown in. Of the 15 headers defined by ANSI C, four are "missing": errno.h, stdarg.h, stddef.h, and locale.h. Actually, errno.h is implemented as error.h. Surprisingly, the old UNIX equivalent of stdarg, varargs.h, is also absent.

It is clear that Lattice sells their compiler to the Asian market since there are switches for handling multibyte character sets for Japan, China and Taiwan, and Korea. However, they have not yet chosen to implement the locale.h header and its corresponding library support. This, along with numerous other library macros and functions, was specifically designed by ANSI to support these environments.

The ANSI-standard headers provided (or their equivalents) contain many extended macros and functions. They are also missing many ANSI macros and functions. The ANSI differences are too numerous too mention here. Suffice it to say that the current release of the library and associated headers clearly does not attempt to be "like ANSI."

Most annoyingly, I found that the standard headers are not precluded from being #included more than once, and they should be. Many programs explicitly #include stdio.h directly and indirectly multiple times. Of course, doing so without protecting your headers results in the generation of many compilation errors. The first thing you should do when you install the Lattice product is put a conditional compilation envelope around each header as follows:

```
/* in stdio.h */
#ifndef __STDIO__
#define __STDIO__
/* distributed header body goes here */
#endif
```

The Bottom Line

Lattice C is quite far from conforming to the proposed ANSI C standard. However, Lattice is not ignoring it—they just seem to be concentrating on OS/2 and other things at present. They have implemented a few choice bits and plan to add a few more in V3.31 soon. You will probably have to wait for V4 (or perhaps V3.4?) to get much more of the standard's goodies. Lattice has their own devoted customer base, who no doubt will be quite happy with the new capabilities this release provides. However, for the time being, I don't see Lattice stealing sales from such companies as Borland and Microsoft, whose implementations are much more modern.

ANSI C Standard Activities

At the September 1988 meeting of ANSI C X3J11, the committee unanimously voted in the draft standard as a formal standard. Unless we messed up administratively or one or more ANSI X3 members vetoes it, the standard should be adopted by ANSI around March or April of 1989. It is hoped that ISO will adopt the same standard later in 1989. The National Bureau of Standards has begun work on a validation suite, and I have agreed to be the liaison on behalf of the ANSI C Standards Committee. ♦

Rex Jaeschke is an independent computer consultant, writer, and lecturer. He is also a member of the X3J11 C Standards Committee, observer on the ISO C Standards Committee, founding editor of The C Journal *and the C language editor of* DEC PROFESSIONAL. *His new book,* Portability and the C Language, *published by Hayden Books in October, is the first text on this popular topic.*

If you would like to comment on, or question any aspect of, ANSI C standard, the validation suite, or the validation process, write to Rex Jaeschke at 2051 Swans Neck Way, Reston, VA 22091, or contact him electronically via UUCP at the following address: uunet!aussie!rex. Rex will endeavor to address your concerns in future columns as space permits.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Smart Terminal Emulation in Windows

by
William S. Hall

Writing a terminal emulator in Windows offers many challenges not normally encountered in the character-based DOS world. But if you can overcome the sometimes surprising difficulties, you can reap many benefits: a rich user interface of dialog boxes and menus, multitasking and data exchange with other applications, the ability to implement special-purpose fonts, and a communications device driver with a complete set of function calls.

About a year ago, I decided to attempt a complete emulation in Windows of the well-known Heath H-19 terminal. The results have been successful, and I'd like to tell you how it was done. I won't overwhelm you with code (although I will use some fragments to illustrate certain points). Rather, I'll emphasize the overall architecture of the emulator, the realization of special features of the terminal, and some of the problems I have encountered and overcome. I'll concentrate primarily on aspects unique to Windows, but I will try to remember that most of you may not have yet acquired an intimate knowledge of the Windows programming environment.

I'll begin by describing the terminal itself. Then I'll discuss an architecture that can be used to implement most of the features of the terminal. Sprinkled throughout will be information about Windows features that must be taken into account in a project of this type.

If you want to see working source, I have provided some extracts illustrating some of the major points to the *PJ* bulletin board. The code consists of two of the five modules, of which the first contains the main window procedures and the second, many of the supporting modules. Emphasis is on those items providing keypad support and communication processing.

The H-19 Terminal

The Heath/Zenith H/Z-19 terminal first appeared about ten years ago. It was amazingly successful, considering that it was created by a rather small company at a time when there were many large and well-known computer terminal manufacturers. A great deal of its success was probably due to its emulation capabilities, its price, and the fact that you could build the terminal from a kit. Thousands were bought by universities, companies, and individuals, and the H-19 proved to be one of Heath's best products. It remains a popular emulation model: The Windows Terminal program contains features of the H-19, MS-DOS Kermit provides H-19 as a terminal type, and although the H-19 is no longer manufactured, it lives on in its successors such as the H/Z-29 and in the Zenith Z-100 computer.

The H-19 is capable of both VT-52 and ANSI emulation and has a so-called Heath mode as well. The Heath mode, in fact, is simply a small but significant enrichment of the VT-52 command set. The Unix operating system recognizes the difference and uses a separate TERMCAPS file. However, if the particular host computer doesn't know about the H-19, you can always set a terminal type of VT-52 with no major loss in functionality. Of course, ANSI mode can also be used. ANSI mode is more or less similar to the VT-100 or VT-102.

The terminal is basic in the sense that it provides a fixed width of 80 columns and a height of 24 lines, but there is also a status line at the bottom that can be accessed only by software commands. As such, it can be used as a protected field. The H-19 also has a set of simple graphics characters. Finally, a character can have only one attribute—that of inverse video.

Every command to the terminal begins with the escape character. In Heath/VT-52 mode, an escape sequence is at most three characters long. For ANSI mode, a sequence is arbitrarily long. The commands are shown in Figure 1.

In addition, the H-19 has a set of special keys that supplement the normal control and alphabetic keys. These include eight function keys that send fixed escape sequences, reset, an on line–off line key, a break key, and a 12-key keypad. The keypad can be used either for numerical entry in normal mode or as a set of editing keys in shifted keypad mode. Finally, in alternate keypad mode, the numerical keys and the associated decimal and Enter keys send out three-character escape sequences instead of numbers.

Almost all these features can be achieved in Windows; however, some are nearly impossible, and some are much harder than others. For example, the first two categories of commands listed above are not difficult. By developing a dedicated font resource, you can achieve the special graphics characters on the H-19. On the other hand, it is hard to turn off key click without specific hardware knowledge of the particular computer, and hold-screen mode is not so natural in Windows as it is on a dedicated terminal. Another problem is making a faithful emulation of the H-19 keypad: The keypad keys in Windows don't follow the same protocol. Finally, not all the keys on an H-19 are found on most computers able to run Windows.

Some Windows Basics

At this point it may be useful to look at the basic layout of a Windows program. Most programs can be abstracted into the following form:

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
WinMain(...) {  /* entry point */

     Init(...);    /* register classes, create
                      windows, init data */
          while (GetMessage(...) { /* read and
                dispatch messages until exit */
               TranslateMessage(...);
               DispatchMessage(...);
          }
     exit();
}

WindowProc1(...) {  /* window procedure for
                       windows of class 1 */
     switch(message) {
          case WM_1:
               f1(...);
               break;
          case WM_2:
               f2(...);
               break;
          .....
     }
}

WindowProc2(...) {  /* window procedure for
                       windows of class 2 */
     switch(message) {
          case WM_1:
               f3(...);
               break;
          case WM_2:
               f4(...);
               break;
          .....
     }
}

f1(...)          /* support functions */
f2(...)
f3(...)
f4(...)
...
```

Essentially, every Windows program remains in the message loop, reading, possibly modifying, and dispatching messages until the program exits. The messages contain input from the mouse and keyboard, requests to redraw the window client area, and other information of various kinds. Windows itself calls the appropriate window procedure, which in turn calls any support functions—local or otherwise—to perform the required tasks. Messages can come from Windows, from the program itself, and from other programs. Typically, a window procedure is a large case statement switching on the message type. Message types are predefined by Windows (WM_CHAR, WM_PAINT, etc.) or user defined by the program (WM_USER, WM_USER+1, etc.). Finally, each message contains one 16-bit and one 32-bit quantity, whose meaning depends on the message type.

Which window gets the message depends on the window's handle. Which window procedure receives and processes the message depends on the window's class. Basically, a window class defines certain attributes shared among all windows of that class, such as the style, window procedure, instance handle, icon, cursor type, background brush, and menu. The class concept abounds in Windows. Several classes, such as edit windows, pushbuttons, listboxes, and static windows, are predefined. Most top-level windows in a program are of a unique class determined by the program itself. In particular, a class can be defined to perform particular tasks by simply specializing its window procedure. You will see how we exploit this idea by defining a smart terminal class in our emulator.

It is worth noting that in our emulator we do not use the message loop described above. The problem is that the **GetMessage** call can't be used to service the communications port. Later, you'll see how this problem can be solved.

Isolating Functionality

A terminal is an I/O device with two sources of input and output. Input can come from either the keyboard or the communications input port. Output can be displayed on the screen or sent to the communications output port. The internal devices can be connected in several ways. For example, when the terminal is off line, keyboard input is displayed on the screen. If on-line, the input to the display comes from the input port and the output port gets its data from the keyboard. Finally, the keyboard may send data to both the display and the port if local-echo is active.

In a Windows emulation of a terminal, the communications device driver provides the port interface, and it is obvious what role the computer's keyboard will play. And, clearly, a window must act as the display screen of the terminal, but what kind of window? The top-level, or main, window in a Windows program can be resized, moved, and suspended (made iconic). It is not a permanently fixed object;

Cursor Functions:
Cursor home, forward, backward, up, down, reverse index, position report, direct addressing, save/restore cursor position.

Erasing and Editing:
Clear display, erase to beginning and end of display, erase line, erase to beginning and end of line, insert and delete line, delete character forward, enter and exit insert mode.

Configuration:
Reset, modify baud rate, enter ANSI mode, enter Heath mode, set and reset status line, key click, hold screen, cursor type, shifted and alternate keypad, auto CR on LF, auto LF on CR.

Modes:
Normal and reverse video, graphics and normal characters.

Other:
Enable and disable keyboard, wrap around, transmit status line, transmit screen.

Figure 1

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

you can imagine the difficulties in performing smart terminal operations in such circumstances. Furthermore, the status line, which on a terminal normally appears at the bottom of the screen, must always be visible if active—even if the window representing the main screen is smaller that the maximum number of lines.

These problems suggest that the screen can be represented by a child of the main window. The child is fixed in size; its actual dimensions in device units are determined by the number of lines and columns in the target terminal and by the font size. The main window thus acts as a viewport into the child and hence of the display screen, as shown in Figure 2.

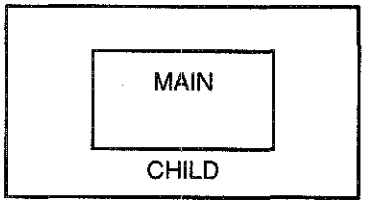


Figure 2

Of course, the main window can be as big as or bigger than the child as well.

Similarly, the status line should also be a child of the main window. Its placement is on the last full line of the main window when it is active. In this way, it will always be visible.

Although child windows can have title bars, borders, and some other attributes, they cannot have menus. So the task of providing the user interface is best left to the top-level window. Such considerations suggest the arrangement of the windows in a terminal emulation program shown in Figure 3.

Once these decisions about the chores of the various windows are made, the modules of the program become more clear. In Windows, it is good practice to isolate functionality into separate segments. Each segment might have one or two entry points and a group of local support functions. The segments are allowed to move, and most are marked as discardable. Here is how I arranged my program:

1. A segment for the main window, which includes the main window function and all functions it calls. In addition, the message reading loop and communications input processor are in this segment. This segment also contains the command processor for any received escape sequences and dispatches all input to the display windows.
2. A segment for the terminal and status windows, which includes their respective Window procedures and all functions called in response to such messages. Hence, all editing and display functions are in this segment. These functions can be reached *only* via the message system.
3. An initialization segment, which can be discarded and is never needed again.
4. A segment containing rarely called functions such as dialog box procedures and which is loaded only upon demand.
5. A resource segment containing error messages and other strings not needed during normal operation as well as icons, dialog boxes, and menus.
6. A data segment containing global variables and the heap with the terminal buffers.

Now let's look at some of the pieces of the program.

The Display Windows

In Windows terminal emulation, it is quite natural to define a class that can perform all the output functions of the target terminal. In the H-19 emulator, I defined a smart terminal class whose window procedure contains many of the features needed for an intelligent display device. The *only* way to gain entry to the window is via the message system. Most of the messages the window can process are user defined. They include

1. Create, display, hide, and destroy the caret.
2. Display and redraw text in the appropriate font and attributes.
3. Maintain the text buffer.
4. Position the caret to any position and return the location.
5. Scroll the display up or down.
6. Insert and delete n characters on a line.
7. Insert and delete n lines of text.
8. Clear screen completely or selectively.

Such a window can serve a variety of emulations. It is only necessary for the command processor to interpret a given escape sequence (or keystroke) as the appropriate message and send it to the window procedure.

Similarly, a window class can be defined to provide status line support. Since it is a single line, the corresponding window procedure just needs to respond to a subset of the above actions.

Because the display windows have a fixed size, it makes sense to talk about a given line or column number, even if that position is not visible in the main client area. On the other hand, several interesting problems can arise. For example, the line on which the caret (the character insert point) is placed must always be visible, so each time the caret moves to another line, a check must be made to see if that line is still visible in the main window. If not, the display window may have to be moved vertically. In turn, if the window is moved, it must be redrawn, which can slow things down. I found that a fast way to perform the positioning is to move the window with no

Main window:
Supports input and communications activities and manages all aspects of the user interface.

Terminal window:
A child of the main window, it is of fixed size and handles all output except for the status line.

Status window:
A child of the main window, it is of fixed size and handles output to the status line.

Figure 3

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

redraw, scroll the display surface accordingly, then update the window. Usually the scroll preserves most of the contents of the window leaving only a small rectangle that must actually be repainted.

The text to be displayed is sent to the display window using a user-defined message. The word parameter is the length, and the long parameter is a pointer to the character string. The display window must not only show the characters, but it must also maintain a buffer of the screen's current contents. Unlike in DOS, where it is enough to place the character/attribute pair into video memory to see it on the screen, the program must also carry out the necessary steps to show the character as well. This can be done as the string is being processed. Or, and aesthetically more satisfying, the characters can be placed in the buffer and the corresponding rectangle on the screen where they will appear can be marked as invalid. A paint message will then be generated with that rectangle as the update region. In response, the paint function for the window draws the new text. Of course, the programmer must write the paint function as well.

The advantage of the show-when-received approach seems to be a smoother, faster display. In addition, control characters, which may generate scrolling of the display and the video buffer, can be processed as received. The advantage of the invalidate-rectangle technique is that text drawing functions can all be in one place.

Another problem comes up in handling the caret. The caret, or character insert point, is a shared resource in windows. Only one program can have the caret. When a window receives the focus for input, a message is sent and the caret can be created. When the focus is lost, the application is again informed and the caret must be destroyed. But smart terminals can be commanded by an escape sequence to change the shape of the caret from, say, block to underline, an action that requires destroying the current caret and recreating it. Since Windows is multitasking, such commands can be received by the display window via the communications port even if the program does not have the focus. Arbitrarily destroying and recreating the caret without regard to ownership is not polite of an application, so care must be taken with this resource.

Since the display window is frequently bigger than the main client area, unless the main client window is exactly an integral number of lines high, a part of the last line can be visible. I solved this problem by setting the size of the status window when inactive to be as high as the remaining visible partial line and placing it just below the last full line on the screen. This window is also always "on top" of the display window. In this way, it defines a clipping region in such a way that no drawing takes place in this "barrier" region. If the status line becomes active, it is increased in height by one line as well so that the text can be displayed on the last full visible line. Arriving at this solution was not an instantaneous process; I tried many other tricks before discovering this one. In addition to greatly improving the display, with this solution the status line is always in the correct position to be seen by the user, providing, of course, that the bottom of the main window is visible.

Off-line Input Processing

Having discussed some of the problems involved in displaying text, we need to look at getting input. Input can come from the keyboard if the terminal is off-line, or from the

Parallel Programming for "C"

INTERWORK™

A Concurrent Programming Toolkit

Interwork is a "C" program library which allows you to write your programs as a set of cooperating concurrent tasks. Very useful for simulation, real-time applications, and experimentation with parallel programming.

FEATURES

- Supports a very large number of tasks (typically more than 100) limited only by available memory. Low overhead per task results in very fast context switching.
- Provides a full set of inter-task communication (ITC) facilities, including shared memory, locks, semaphores, blocking queues, and UNIX™-style signals. Also has building blocks for constructing your own ITC facilities.
- Handles interrupts (DOS version) and integrates them into task scheduling. Supply your own interrupt handlers or block tasks on interrupts.
- Lets you trace task switches and inter-task communication.
- Comes with complete documentation including a user's manual and reference manual of commands.

Interwork is available for the following systems:

Hardware	*Operating System*	*Price*
IBM PC, XT, AT	PC-DOS 2.0 or later	$129
IBM PC AT	XENIX™	$159
DEC VAX™, SUN	UNIX 4.2BSD	$249

PC-DOS version is compatible with DeSmet, Lattice, and Microsoft C compilers.

Please specify hardware and operating system when ordering. Shipping and handling included; COD orders add $2.50. Send check or money order to:

Block Island Technologies

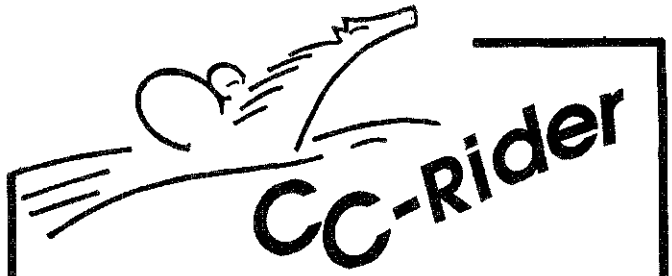
Innovative Computer Software

13563 NW Cornell Road, Suite 230, Portland, Oregon 97229-5892

(503) 241-8971

Trademarks: Interwork, Block Island Technologies; UNIX, AT&T Bell Laboratories, Inc.; XENIX, Microsoft, Inc.; VAX, Digital Equipment Corporation

Number **224** on the Reader Service Card



POP UP YOUR SYMBOLS

Get *Tansi* by adding ***hypertext*** capabilities to your C editing environment. Gain access to all your macro, function, structure and typedef definitions instantly!

See CC-RIDER

SEE WHAT YOU CAN DO!

INCLUDES:

CCSYM Standard C "symbol compiler" which builds the symbol database used by CC-RIDER. Supports Microsoft C v5.1 extensions.

CCRIDER Pop-up program to recall your symbol definitions in a special window from any program.

USE YOUR OWN EDITOR

Western Wares

Box C Norwood, CO 81423

(303) 327-4898 ***Demo $5***

C.O.D., VISA, MASTER CHARGE and AMEX

ONLY $89

Number **100** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

communications port if on-line. It may be a bit simpler to understand the process if we look first at the case when the terminal is off-line. In this case, the terminal acts as a typewriter.

Windows provides an asynchronous way of receiving character input. Whenever a key is typed, three messages are generated—WM_KEYDOWN, WM_KEYUP, and WM_CHAR. The first indicates that the key has been depressed and the second, that it has been released. Finally, if the character is printable, the virtual key code, which is part of the keydown/keyup message structure, is realized as an ANSI character in the WM_CHAR message.

In our program, all input is handled by the main window procedure. When that window has the focus, keyboard messages can be received. If the terminal is off-line, printable and control characters are shipped as one-character strings to the appropriate child window. The main window maintains an active handle variable and simply sends the message to that window. If the status line is the active window, the character will appear there; otherwise, the display window will receive the character. (The status window becomes the active window only if it has been turned on by the appropriate escape sequence and if a command sequence has been received to move the caret to the status line.)

If the escape character is typed, the state of the command processor changes, and the appropriate number of characters that follow are processed as a command. If the command affects the display, such as movement of the caret, the corresponding message is sent to the active window; otherwise, it is handled locally, as is, for example, a sequence to change the baud rate. If an edit key is struck (one of the keypad keys, for example), WM_KEYDOWN message is translated into an editing command for the display and the corresponding message is sent to the active window.

Edit keys, such as cursor motion keys, are interpreted as commands for the displays, and appropriate messages are sent to the active display window. Finally, certain function keys can have an effect on the terminal. For example, F6 has been dedicated to mimic the H-19 CLEAR key.

The actual processing of editing keys is very complicated, since the goal was to emulate the exact action of the H-19. We will discuss this some more later.

Modifying the Message Loop

It is easy to get character output to the communications port when on-line; whenever a WM_CHAR message is received from a call to **GetMessage**, write the character to the port. The problem is how to know when there are characters to be read; no message is sent when the communications port read buffer is not empty; instead, the communications driver must be polled. But if the program waits on the port and does not read messages, all other Windows programs will stop until the next **GetMessage** call.

There is a way around the problem. It is possible to replace the **GetMessage** call with **PeekMessage**. When **PeekMessage** is called, the program yields to all other tasks, which then execute in turn. Control then returns to the program, even if there is no message. So tasking and polling can be maintained with the following kind of message loop:

```
while(TRUE) {
    if (PeekMessage(...)) {
        if (message == WM_QUIT)
```


C™ Programmer's Toolbox

PC/MS DOS 2.1 & later

Volume I

CFLow™ — Determine your C program's function hiearchy; external P/T functions; etc.

CPrint™ — Beautify C source code to your selected format

PMon™ — Monitor your program/OS execution; Determine where time is being spent...

+ 12 more tools...

Are you creating better, faster, higher quality and easier to read programs in a fraction of the time?

Join our many beginning and expert users.

Volume II

9 more tools that you requested including:

CLint™ — Perform additional syntax checks of your C source code-find silly mistakes...

CritPath™ — Determine your C program's critical path from PMon and CFlow report inputs...

CXRef™ — Trace your symbol/variable usage; identify potential usage problems...

The Toolbox works with all compilers and libraries. Specify any number of files through wildcards and/or indirect files.

Buy both Volumes I & II for $140, saving $20, or only one volume for $79.95. CA residents please add applicable tax. Overseas, please add $10 shipping & handling. With our 30 day money back guarantee, you can't lose.

Call or write us today

MasterCard VISA®

C™ Programmer's McTool Series

Apple® Macintosh™

Common Tool Features
- Powerful, built-in editor
- Automatic/manual search & replacement
- Selective C source code highlighting
- Works with all C compilers
- ANSI support

#1 McCPrint™ $59.95
#2 McCLint™ $99.95

MMC AD Systems
Box 360845
Milpitas, California 95035
(408) 263-0781
'The C Tool Specialists'

Number 94 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
                exit();
            else {
                TranslateMessage(...);
                DispatchMessage(...);
            }
        else
            ProcessComm();
    }
```

PeekMessage returns non-zero if a message is read, and it is dispatched the same as before. If there are no messages, however, the communications buffer can be polled in **ProcessComm** and any characters present can be read and dispatched to the command processor of the emulator.

Window Subclassing and On-line Processing

When a terminal goes on-line with no local-echo, the keyboard and screen are disconnected from each other and are connected to the communications port device. If local-echo is on, the keyboard remains connected to the screen as well.

Similarly, in the emulator the action taken upon receipt of the WM_CHAR, WM_KEYDOWN, and WM_KEYUP messages changes. Characters are now routed to the communications port, and edit keys send the expected escape sequences to the host computer. If local-echo is also in effect, however, the keyboard must also act as if it were connected to the display. Such action can be easily achieved by a technique known as *subclassing*. Here is how it works.

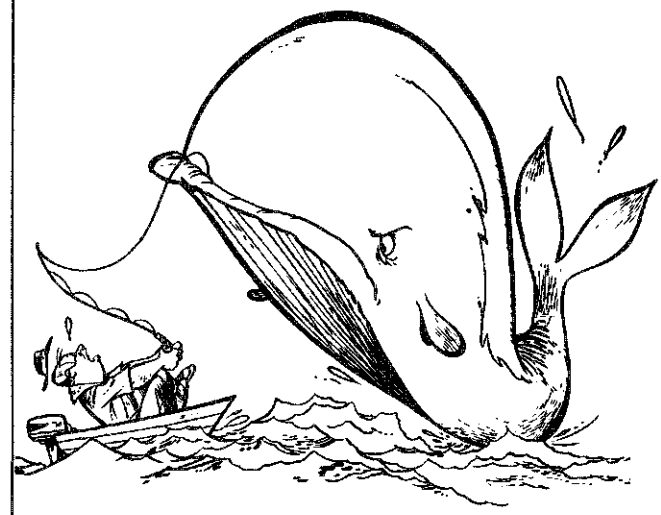
Suppose the main window has a procedure that looks something like this:

```
MainWndProc(...) {
    switch(message) {
        case WM_CHAR:
            DisplayCharOnScreen();
            break;

        ....

        default:
           return(DefWndProc(...));
    }
    return (0L);
}
```

We now write another one:

```
SubClassWndProc(...) {
    switch(message) {
        case WM_CHAR:
            WriteCharToPort();
            if (!local_echo)
                break;

        default:
            return(CallWndProc((FARPROC)
                   MainWndProc(...)));
    }
    return (0L);
}
```

When the terminal is placed on line, the main window's window procedure is changed to point to the subclass procedure, which can be done with a single call to Windows. WM_CHAR messages are now routed to the communications port, and all other messages are simply passed on to the main window procedure. But if local-echo is on, WM_CHAR messages also go on to the main window procedure, since the case statement now falls through to the default action.



Is your C program too much to handle?

You need PC-lint 3.0

PC-lint will analyze your C programs (one or many modules) and uncover glitches, bugs, quirks, and inconsistencies that your compiler, working on one module at a time, will miss.

PC-lint will find uninitialized variables, function inconsistencies, argument/parameter mismatches, wild pointers, inconsistent declarations, unaccessed variables, suspicious macros, unreachable code, indentation irregularities, unusual expressions, format irregularities, and ... **New with 3.0** array and `struct` access and initialization, `case` fall-through, precision tracking, complete `const` checking and more. Support for Microsoft 5.1, Turbo 2.0, Watcom.

Customize PC-lint to your own programming style and project. Turn off any error message locally or globally, alter size of scalars, adjust format of error messages, turn off errors on the basis of symbol name and more.

Mainframe and mini programmers -- Use Generic Lint in shrouded source form on VAX/VMS, Unix, Xenix, Microport, OS-9, IBM VM/CMS - MVS, AOS/VS etc. Requires only K&R C to compile. Prices start at $798. Call for details.

(215)584-4261

Only $139 - CALL TODAY
30 Day Money-back Guarantee

Gimpel Software

3207 Hogarth Lane Collegeville PA 19426

Works with any compiler - Specify MS-DOS or OS/2
Quantity and educational discounts available.
PA orders add 6% sales tax -- Outside USA add $20
PC-lint and Generic Lint are trademarks of Gimpel Software.

Contact this advertiser directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Of course, we use subclassing for more than just handling the WM_CHAR message. It is a powerful technique. Subclassing gives the programmer the ability to alter the behavior of any window whose handle and instance handle can be obtained. This includes the predefined ones, such as dialog boxes.

Special Keys

You may have observed the call to a perhaps mysterious function, **TranslateMessage**, in the message loop. This windows function simply converts WM_KEYDOWN and WM_KEYUP messages from Windows into WM_CHAR messages if a printable character for the keystroke exists. By controlling and modifying the keystroke messages and their translations in the message loop, I was able to achieve a precise emulation of the H-19 keypad as well as other special keys.

The H-19 keypad appears as follows:

IC(7)	**Up(8)**	**DC(9)**
Left(4)	**Home(5)**	**Right(6)**
IL(1)	**Down(2)**	**DL(3)**
0	**.**	**?**

The ? means that we do not have the extra Enter key on IBM PC 83-key keyboards, and at this time, we have made no effort to emulate it.

The operation of the keypad on the H-19 requires some explanation. Generally, it can be in normal, normal shifted, alternate, and alternate shifted modes. Each one of these modes can be set with an escape sequence. The output from a keypad key depends on the mode. For example, if we press the Up arrow key in normal mode, the digit 8 is sent. But if we press the Up arrow in alternate keypad mode, the sequence Esc ? x is transmitted. These actions are further modified if the Shift key has been depressed.

To emulate this action in Windows, it is necessary to translate the virtual key codes from the keypad to the ones desired. Furthermore, it is sometimes necessary to inhibit translation to WM_CHAR messages. Continuing the example above, if we are in normal keypad mode and we receive the VK_UP message, we have to convert this to the corresponding VK_NUMPAD message—in this case, VK_NUMPAD8. Then, since we want the key to be displayed as an 8, we let translation take place. If we are in alternate keypad mode, we make the same conversion without allowing translation. In this case, only a WM_KEYDOWN and WM_KEYUP message will be seen, but the WM_CHAR will not. When the WM_KEYDOWN message is received, we can then send out the Esc ? x sequence.

Similarly, conversion of VK_NUMPADn messages to the corresponding edit keys may be needed, depending on the settings. All together, there are 8 combinations.

To complicate the issue, the position of the PC Num Lock key will modify the messages expected. As implemented, the keypad works correctly with this key in either position; this was rather difficult to achieve.

A similar problem is the lack of a break key in Windows. In this program, Ctrl-Break is used. Normally, this key is translated to Ctrl-C. By inhibiting translation, we can use the keydown message as a signal to turn break on, and the keyup message to turn break off. At the same time, we are assured that Ctrl-C is not also transmitted.

You might want to compare what we have done with the use of accelerators in Windows, where the accelerator keystroke not only is not translated, but it is also not even dispatched directly—it is converted to a WM_COMMAND message. In our case we let it go through, but we may modify its value, and we control its translation.

Miscellaneous Features

The H-19 terminal has two dip switches that determine the power-up state of the machine. The analogy in Windows is the **win.ini** file. When the emulator is first run, it reads **win.ini** and sets initialization parameters accordingly. On the terminal, all parameters except local-echo mode can be changed as needed by going off line and typing in an escape sequence. This is still possible on the emulator, but it is easier to use the menus and dialog boxes provided. In addition, settings can also be written to **win.ini**.

The H-19 character set is 7-bit ASCII. The characters are 7 bits by 9 bits in an 8-by-10 cell. In addition, there are 33 special characters that provide limited graphics capability. Using the Windows font editor, I was able to build a font resource in which the graphics characters are identical to the original. Note, however, that the normal characters had to be weighted a bit more heavily in the vertical direction to achieve a good appearance on the white Windows screen. The graphics characters occupy the otherwise empty positions of the control characters, which may explain why exactly 33 were provided on the H-19.

Each character can have the attribute of inverse video. The inverse video attribute is stored in the video buffer with the eighth bit set. When the character is displayed, the screen and background colors are interchanged just before the bits are written to the screen.

Some features of the terminal were rather difficult to implement. One of these was hold-screen mode, where a line (screen) is displayed and then the display stops scrolling until Scroll Lock (Shift–Scroll Lock) is pressed. Part of the action requires dynamically modifying the state of the Scroll Lock key, which is possible in Windows 2.0. Unfortunately, the Scroll Lock light does not follow the state on AT and enhanced keyboards.

Another problem was transmitting the contents of the status line or the entire screen. Each group of characters in the screen buffer must be sent along with the commands that set its attributes. For example, if a set of characters is graphics and in reverse video, the sequences to put the terminal into reverse video and to set graphics mode must precede the characters. Ideally, if an H-19 screen is sent to another H-19, the appearance should be the same on both. To this end, I allocate space in the Windows global heap and move the characters there while determining the necessary escape sequences. Then they are read back to the communications port a line at a time whenever Windows returns to the emulator from **PeekMessage**. In this way the screen or status line can be transmitted while other Windows programs continue to run.

Finally, let me mention that text data can be "pasted" from the Windows clipboard. We thus have a rudimentary file transfer, although it is not very reliable if the host computer is easily overwhelmed by too many characters at once. *Pasting*, in this case, means actually sending the characters as if they were being typed at the keyboard. As they echo back on the screen, it appears as if they were actually placed there rather than simply sent to the communications port. The techniques used in sending a line or screen were useful here, although it is not necessary to format the text, since we assume that no special H-19 character attributes need to be supplied.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Microsoft C Users

Lattice C Users

Frustrated by Long Development Cycles? Constrained by 640K Limitations?

"It's the best of both worlds for the experienced C programmer."
- PC Magazine, September 13, 1988

"...the best product which has ever been written for next generation PC's."
- David Kuechler, Vice President R&D, Program Technologies

Instant-C/16M™ is your solution.

"Unique and highly responsive C programming environment, ... ideally suited for developing C application prototypes."
- PC Week, August 1, 1988

"...best program development environment available anywhere."
- Karim Lakhani, Vice President Technical Development, Fast Trax

Instant-C/16M is the perfect complement to the Microsoft C and Lattice C compilers. You can develop your Microsoft C or Lattice C program in *Instant-C/16M* and save more than half of your development time. Once your program has been completely debugged with *Instant-C/16M*, compile it with Microsoft C or Lattice C for smallest code size and optimal code performance. You get the best of both worlds: fast development and superb debugging from *Instant-C/16M* and fast code from your current C compiler. Development benchmarks consistently show that *Instant-C/16M* is at least **twice as fast** as either Turbo C 1.5, Microsoft C 5.1, or QuickC 1.0 when comparing the turnaround times for compiling and linking.

Instant-C/16M was designed to give the professional programmer the ability to **make much larger programs -- much faster.** By providing access to the full 16MB of extended memory on 80286/386-based PCs, *Instant-C/16M* lets you create any size program you wish with all the features you want. ***Instant-C/16M* is simply the best and fastest way to develop and debug your C programs.**

Instant-C/16M

Instant-C/16M is constructed around an incremental C compiler and includes: a source-level debugger, editor, run-time checker, and library source. ***Instant-C/16M*** lets you add all the features you want to your program without worrying about the 640K limit. Programs of 75,000 lines of code and more than 1MB of dynamically allocated memory are easily handled.

Instant-C/16M's debugging capabilities are unsurpassed. Programs too large for Codeview™ fit easily within ***Instant-C/16M*** to make debugging large programs much easier. Combine ***Instant-C/16M***'s automatic run-time checking with protected mode hardware's checking and you have a "safety net" for catching those particularly difficult bugs that might normally take you hours, days, or even weeks to find and correct.

Features and Benefits

- ***Large Program Development*** -- *Instant-C/16M* gives you access to 16MB of extended memory on 80286/386-based machines. Development of multi-megabyte programs under DOS is no longer a problem. -- **Develop and debug very large programs.**
- ***Instant compilation*** -- Whenever you make a change *Instant-C/16M*'s incremental compiler recompiles only the changed portions of your code, not your entire source file. A typical change takes only one half (.5) to 2 seconds. -- **Saves your valuable development time.**
- ***Instant linking*** -- *Instant-C/16M*'s incremental compiler automatically links your code. -- **Zero link time.**
- ***Integrated environment*** -- The compiler, debugger, editor, run-time checker, and source formatter are tied together. You no longer need to quit one tool to start up another one. And all of *Instant-C/16M*'s tools are just a single keystroke away. -- **Everything is in one package.**
- ***Superior debugging*** -- Source-level debugger automatically traps errors in: invalid pointer references, computed function addresses, overstored return addresses, out of bounds array references, invalid pointer differences, conversion errors, invalid shifts, and arithmetic exceptions. The debugger supports the full C language including local symbols and #defines. -- **The best debugger available for C today.**
- ***Supports your existing programs*** -- *Instant-C/16M* supports the Microsoft and Lattice C compilers and all of your current C source code and libraries. -- **Protects your software investment.**

Instant-C

Instant-C has all the same productivity benefits of *Instant-C/16M* and supports 8086 and 8088-based PCs and compatibles.

Instant-C is designed to run your program at speeds comparable to C compilers. In addition, it lets you generate stand-alone programs -- if you don't need the benefits of a traditional optimizing compiler, don't buy one. ***Instant-C*** can be the only development tool you use.

Just like *Instant-C/16M*, ***Instant-C*** offers you full support for Microsoft C 5.1 and Lattice C 3.2 and support for graphics boards and non-PC hardware.

Why waste any more of your development time? ***Instant-C*** lets you develop your code twice as fast.

The Optimal C Development System

Develop and Debug
in
Instant-C/16M
then
Deliver
with
Your Favorite C Compiler

Unconditional 30 Day Money Back Guarantee!

Trademarks are acknowledged to Rational Systems, Inc., Microsoft Corp., Lattice, and Borland.

For More Information Call or Write

Rational Systems, Inc.
P.O. Box 480
220 North Main Street
Natick, MA 01760
Tel: **(508) 653-6006**
FAX: **(508) 655-2753**

Number 20 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Performance

Windows has been criticized for not being a good real-time system for rapid display of text. Under Windows 2.0, however, text display has been greatly improved despite the fact that the Windows screen is always in a high-resolution graphics mode. I have found that with proper flow control (XOFF- XON, for example), communications at 9600 baud are quite satisfactory on the EGA, VGA, Hercules, 8514A, or the Olivetti/AT&T 640x400 monochrome displays.

Conclusion

In this project, I have made strong use of such Windows features as the menu/dialog box user interface, the rich set of communications functions so lacking in DOS, and the ability to use special fonts. I have even been able to base a Windows version of Kermit on the emulator. Even so, I've exploited only a small part of Windows' real capability. Yet to appear are emulators of some of the popular graphics terminals, but that is a natural application of Windows. I hope the information here may point you in that direction if you are so inclined and that it has, at the very least, acquainted you with the special considerations that are necessary in the Windows environment. ♦

As a Senior Engineer at Olivetti Advanced Technology Center in Cupertino, California, Bill Hall specializes in Windows system support software.

Code follows.

CBTREE, the easiest to use, most flexible B+tree file manager for fast and reliable record access

CBTREE... Includes over 8,000 lines of 'C' source code FREE! Since 1984, thousands of 'C' programmers have benefited from using CBTREE.

Save programming time and effort. You can develop your applications quickly and easily since CBTREE's interface is so simple. You'll cut weeks off your development time. Use part or all of our complete sample programs to get your applications going FAST!

Portable 'C' code. The 'C' source code can be compiled with all popular C compilers for the IBM PC including Microsoft C, Quick C, Turbo C, Lattice C, Aztec C and others. Also works under Unix, Xenix, AmigaDos, Ultrix, VAX/CMS, and others. CBTREE includes record locking calls for **multi-user and network applications.** The CBTREE object module is only 22K and is easy to link into your programs. You don't even pay runtime fees or royalties on your CBTREE applications!

Reduce costs with system design flexibility. CBTREE allows unrestricted and unlimited designs. Reduce your development costs. You define your keys the way you want. Supports any number of keys, variable key lengths, concatenated keys, variable length data records, and data record size. Includes crash recovery utilities and more.

Use the most efficient search techniques. CBTREE is a full function implementation of the industry standard B+tree access method, providing the fastest keyed file access performance.

Database Calls:

- Get first
- Get last
- Get previous
- Get next
- Get less than
- Get less than or equal
- Get greater than
- Get greater than or equal
- Get sequential block
- Get partial key match
- Get all partial matches
- Get all keys and locations
- Insert key
- Insert key and record
- Delete key
- Delete key and record
- Change record location

CBTREE is only $159 plus shipping - a money-saving price!

We provide **free telephone support** and an **unconditional 90-day money back guarantee!**

To order or for additional information on any of our products, call **TOLL FREE 1-800-346-8038** or **(703) 847-1743** or write using the address below.

NEW CBTREE add-on products to make your programming easier. AVAILABLE NOW! • Adhoc query, reporting system; • SQL interpreter.

If you program in 'C', sooner or later you're going to need a B+tree. Don't delay until you're in a crunch, plan ahead, place your order for CBTREE now. Orders shipped within 24 hours!

PEACOCK SYSTEMS, INC.

PEACOCK SYSTEMS, INC.
2108-C GALLOWS ROAD
VIENNA, VA 22180

Number **216** on the Reader Service Card

JAKE™: A NATURAL LANGUAGE SOFTWARE BREAKTHROUGH

Use **JAKE** to create a natural language front end to your database, game, or graphics program. **JAKE** allows the ultimate in user friendliness: users simply type commands and questions using their own preferred vocabulary. **JAKE** offers your users a high degree of interactivity, virtually instant access to information, a dramatic reduction in training time, and subsequent ease of use.

JAKE is an application-independent library usable with Microsoft and Turbo C for translating English commands and queries into function calls and data structures. **JAKE** offers context-sensitive semantic processing, interfaces easily to any application, and uses <64K of memory. $495 complete.

Sound too good to be true? Get our interactive demo for only $10 and see.

CALL (408) 438-6922 VISA, MC

EKS English Knowledge Systems, Inc.
5525 Scotts Valley Dr. Suite 22
Scotts Valley, CA 95066

Number **243** on the Reader Service Card

— SUBEXPRESSION —

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

All code in this issue is available on 360K, MS-DOS disk.
ORDER BY PHONE — VISA/MC orders call toll free 800/234-0386.
Mail order information on pages 8 and 88.

Code from "Smart Terminal Emulation" by William S. Hall

1

```
/* Windows H19 emulator
 *
 * Written by William S. Hall
 *           3665 Benton Street, #66
 *           Santa Clara, CA 95051
 * Program entry module and Emulator function support module-extracts
 */

/* Windows entry point */
int FAR PASCAL WinMain(hInstance, hPrevInstance, lpszCmdLine, cmdShow)
HANDLE hInstance, hPrevInstance;
LPSTR lpszCmdLine;
short cmdShow;
{
   /* Initialization */
    if (!InitProgram(hInstance,hPrevInstance, lpszCmdLine, cmdShow))
        return FALSE;

  /* Message loop */
    while (TRUE)
        if (!DoMessage()) /* if message, process it */
            ProcessComm();    /* if none, then take care of communications */
}

/* Message processing loop.  Return false if no message. */
BOOL NEAR DoMessage()
{
    MSG msg;

  /* Look for a message.  If we have one, remove and process it. */
    if (PeekMessage((LPMSG)&msg,NULL,0,0,PM_REMOVE)) {
      /* have to exit. */
        if (msg.message == WM_QUIT)
            exit((int)msg.wParam);
       /* got a key from accelator table.  All shifted, control,
        * and shifted-control function keys are in this table.
        * We convert them to a WM_COMMAND message and process them there.
        */
        if (TranslateAccelerator(MW.hWnd,hAccel,(LPMSG)&msg) == 0) {
           /* DoKeyTranslation is our own function.  Here we
            * decide if a key down/up message should be translated to
            * a WM_CHAR message, or if the wParam should be changed
            * to another virtual key code.  If true is returned, the
            * key is translated.
            */
            if (DoKeyTranslation(msg.message, &msg.wParam))
                TranslateMessage((LPMSG)&msg);
            /* Tell Windows to call our message procedure. */
            DispatchMessage((LPMSG)&msg);
        }
        return TRUE;
    }
    return FALSE;
}

/* This complicated routine reads the comm port when on-line provided
 * the buffer is empty.  Otherwise, it looks to see if anything
 * is left over from a previous attempt at processing.  Normally,
 * the buffer is completely emptied when it is sent to the main
 * window procedure.  However, if hold-screen is in effect, then
 * the terminal window will stop display whenever a carriage return
 * is seen.  In this case, the remaining buffer remains unprocessed
 * until the scroll-lock key is toggled and another line (or screen)
 * can be displayed.
 */
void NEAR ProcessComm()
{
    COMSTAT ComStatus;
    int num;
    int retresult = 0;
    int result = 0;
    static char Buffer[COMMQUESIZE];
    static int bufsize = 0;

    /* if on line, continue */
    if ((CD.LineState == IDM_ONLINE) && !CD.ScrollLock) {
      /* if the buffer is empty, read the comm port */
        if (bufsize == 0) {
            GetCommError(cid, (COMSTAT FAR *)&ComStatus);
            /* if we find something in the comm buffer, get it */
            if (num = ComStatus.cbInQue) {
                num = min(num, BUFSIZE);
                /* if error, ignore and pass the string along anyway */
                if ((result = ReadComm(cid, (LPSTR)Buffer, num)) < 0) {
                    result = -result;
                    Buffer[result] = NUL;
                }
            }
        }
        else
            result = bufsize;
      /* We got something on the last read, or the buffer was not empty
       * because the previous dispatch did not read all of it.
       */
        if (result) {
            retresult = H19StringInput(Buffer, result);
            /* if still not empty, move remaining to front of buffer */
            if (retresult) {
                bufsize = retresult;
                memmove(Buffer, Buffer+result-retresult,retresult);
                Buffer[bufsize] = 0;
            }
            else
                bufsize = 0;
        }
      /* If nothing immediate to process, then check out global buffer
       *   in case something from the clipboard is there.
       */
        else if (GB.hBuf) {
            BYTE FAR *tbuf;
            LONG BufBytesRemaining;
            int count;

            /* if not empty, then get another line from buffer */
            BufBytesRemaining = GB.lBufTail - GB.lBufHead;
            if (BufBytesRemaining > 0) {
                tbuf = GlobalLock(GB.hBuf) + GB.lBufHead;
                if (BufBytesRemaining > INT_MAX)
                    count = TW.MaxCols;
                else
                    count = min(TW.MaxCols, (int)LOWORD(BufBytesRemaining));
                /* send it to the port */
                WriteToPort(cid, tbuf, count);
                GB.lBufHead += count;
                BufBytesRemaining = GB.lBufTail - GB.lBufHead;
                GlobalUnlock(GB.hBuf);
            }
            if (BufBytesRemaining <= 0)
                GB.hBuf = GlobalFree(GB.hBuf);
        }
    }
}
```

2

NOTE: *This is only a sample portion of the code available for this article. Complete listings are included on this issue's code disk or from our BBS services. See pages 8, 88 & 95 for more information.*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

COBOL Comes of Age

by
Richard Riehle

COBOL was introduced in 1959, before we had formal notions of "structured programming" or "top-down design." Early COBOL programmers were practical people with no training in computer science theory and very little precedent. However, they did get the job done in industries as diverse as insurance, banking, automotive manufacturing, retailing, and public utilities. Consequently, COBOL remains the language of choice for much business data processing—but not without progress *and* change.

For starters, computer science graduates from universities in the 1970s used Pascal or PL/1 as their principal language. In response, many of the graying, '60s programmers—some now elevated to loftier management positions, such as MIS directors and programming managers—began to dig in their heels, resisting any idea of improving their beloved COBOL. Still, the Pascal invasion was inexorable; devotees of structured programming began to exert greater and greater influence. Next, CODASYL (Committee on Data Systems and Languages) in 1981 proposed a new COBOL standard based on radical new ideas of programming . . . well, ideas that seemed radical to many old-guard COBOL practitioners.

Present-day users of Pascal, Modula-2, Ada, and C will find familiar ideas in COBOL-85. New features include in-line loops (equivalent to "while ... do" and "repeat ... until"), scope terminators for conditional statements ("if ... end-if"), local and global variable declaration, called subprograms, a case statement, parameter passing between called procedures, and other features we have come to accept as de rigueur in most modern languages. Some of you might look back over the preceding list and ask, "How could they even write programs at all?"

Let's take a look at some of the new COBOL-85 constructs. We shall make most of our comparisons to COBOL-74, the previous ANSI standard for COBOL.

Iteration

Most production programs are fundamentally large loops containing many smaller loops. In "traditional" COBOL, one of the main looping constructs has been the PERFORM. The syntax (simplified) has been

```
PERFORM procedure name UNTIL condition
```

Somewhere in a COBOL program a procedure must exist that can be referenced by the PERFORM. The COBOL-85 enhancement permits an in-line loop, delimited by a scope terminator (END-PERFORM), that looks like this:

```
PERFORM UNTIL condition
   some procedural statements
END-PERFORM
```

All the statements between PERFORM and END-PERFORM will be executed exactly as if they had been coded in a procedure in some distant part of the program.

But this type of loop is still very limited. The developers of the COBOL-85 standard decided to provide a PERFORM variation that behaves like the well-known "while ... do" loop. The syntax is

```
PERFORM
   WITH TEST BEFORE
        UNTIL condition
   some procedural statements
END-PERFORM
```

The phrase WITH TEST BEFORE indicates that the loop should not be performed unless the condition tests true prior to each pass. In place of WITH TEST BEFORE, the programmer may use the expression WITH TEST AFTER, which forces the loop to complete at least one pass before terminating.

Users of modern programming languages take in-line loops for granted. Some old-time COBOL programmers fail to see the advantage of being able to do a PERFORM without specifying a procedure to PERFORM. However, the advantage of a "while ... do" construct clicks into place fairly quickly with most experienced COBOL programmers.

Selection

Standard COBOL has had one fundamental statement for selection: "IF ... ELSE"—the subject of controversy and argument in COBOL circles. You would think every COBOL veteran would welcome some improvement; many have, but some still miss the point. In traditional COBOL, we have

```
IF condition  [ do something ]
ELSE          [ do something else ]
```

Sometimes we have nested statements:

```
IF condition-1          NOTE: There is no
     [do A]                   "do" statement
   IF condition-2             in COBOL.
      IF condition-3
         [do B]
      ELSE
         [do C]
   ELSE
      [do D]
ELSE
    [do E].
```

The nested IF has created fascinating maintenance problems—even the original coder sometimes is unsure about what was really intended. Some COBOL programming managers simply prohibit the use of nested IF statements. Others complain that one cannot avoid using the evil GO TO unless one uses nested IFs. Attempts to correct errors resulting from nested IF statements often propagate still more errors.

To remedy this situation, COBOL-85 has introduced the previously mentioned scope terminator, which is used for all

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

conditional statements in COBOL-85 programs. For an IF statement in COBOL-74, we have

```
IF condition THEN
    [do something]
ELSE
    [do something else]
END-IF
```

In COBOL-85, we would have

```
IF condition-1
     [do A]
     IF condition-2
         IF condition-3
             [do B]
         ELSE
             [do C]
         END-IF
     ELSE
         [do D]
     END-IF
ELSE
      [do E]
END-IF
```

The difference is subtle—but significant—when examined in the context of the elementary structures described by Jacopini and Bohm (see sidebar, "The Structure Theorem"). Experienced COBOL programmers will notice that I omitted the period after the final END-IF. In COBOL-85, the scope terminator is used instead.

Other elementary conditional COBOL statements have also been updated. One of the most significant changes concerns the READ statement. Prior to COBOL-85, READ was coded as

```
READ file-name
     AT END
         [do something].
```

With COBOL-85, we have

```
READ file-name
     AT END
         [do something]
     NOT AT END
         [do something else]
END-READ
```

Again, notice the elimination of the delimiting period found in prior incarnations of COBOL. The above statement might be coded as follows:

```
READ ACCOUNTS-FILE
     AT END
          PERFORM LAST-ACCOUNT-RECORD
     NOT AT END
          ADD 1 TO ACCOUNT-RECORD-COUNT
```

The Structure Theorem

Before the advent of "structured programming," certain styles of program organization were dubbed "spaghetti code" because of the many threads of logic in the code and the manner in which it often doubled back upon itself. Debugging such code was like trying to remove the sauce without disturbing the noodles.

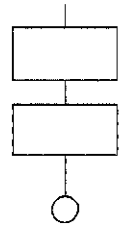
A solution to this problem came in 1966, when two European computer scientists, Bohm and Jacopini, published a paper proving that any correct program could be constructed using elementary structures. Their work is often referred to as the "structure theorem." Written in Italian, the paper remained unread in the English-speaking world for a long time, only beginning to receive accolades for its brilliance in the mid 1970s. Even now, though, most people who talk of structured programming do not know of the supporting formal proof.

The current literature on the structure theorem describes three elementary structures: Sequence, Iteration, and Selection. Sequence is a simple list of actions. Iteration is a loop, and Selection is a decision made on the basis of testing for some condition or conditions. The diagrams below illustrate each of these.

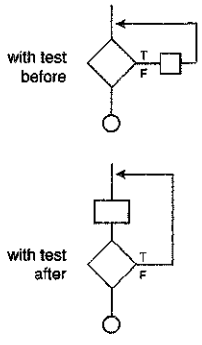
COBOL 85 has been updated to include each of these structures and make it possible to write correct programs on the basis of the structure theorem. This is one of the most important enhancements to COBOL since it was originally designed.

One special note: The little circle in each diagram represents a "scope terminator" and is implemented in COBOL-85 via the "END-" verb associated with most of the COBOL verbs. The scope terminator is an important element in correctly coding these structures.

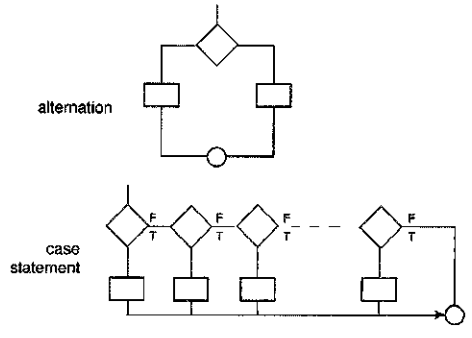
It seems especially fitting that two Italian computer scientists should have made such a significant contribution to the unraveling of spaghetti code. ♦



Sequence



Iteration



Selection

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
          PERFORM ACCOUNT-UPDATE-ROUTINE
END-READ
```

If we were to enclose this statement in an in-line PERFORM as described above, we might have

```
001-MAIN-ROUTINE.
     PERFORM UNTIL END-ACCOUNTS
        READ ACCOUNTS-FILE
            AT END
                  PERFORM 005-LAST-ACCOUNT-RECORD
            NOT AT END
                  ADD 1 TO ACCOUNT-RECORD-COUNT
                  PERFORM 002-ACCOUNT-UPDATE-ROUTINE
        END-READ
                ... more program statements ...
     END PERFORM
     STOP RUN.
002-ACCOUNT-UPDATE-ROUTINE.
                  :
                  :
005-LAST-ACCOUNT-RECORD.
```

Another variation on the "selection" is the CASE statement. In COBOL-85 it is called the EVALUATE statement. Traditional COBOL has the following primitive implementation of the CASE statement:

```
GO TO ... DEPENDING ON variable.
```

When the GO TO fell into disrepute, so did GO TO ... DEPENDING. Moreover, GO TO ... DEPENDING was useful only for numeric data, much like the ON ... GOTO of BASIC. The EVALUATE statement has the following form:

```
EVALUATE [data name or statement]
    WHEN [condition]
         [do something]
    WHEN [condition]
         [do something]
    WHEN OTHER
         [do something]
END-EVALUATE
```

This construct is frequently used in place of the nested IF. Although EVALUATE seems to create less confusion than nested IF statements, some programmers create problems for themselves by not remaining alert to the scope rules. EVALUATE statements may also be nested, however, and that is where some of the problems occur.

Here is an example of EVALUATE within an in-line PERFORM:

```
 PERFORM UNTIL END-OF-FILE
    READ INVENTORY-MASTER
         AT END
            PERFORM END-OF-INVENTORY-MASTER
         NOT AT END
            ADD 1 TO INVENTORY-MASTER-RECORD-COUNT
            EVALUATE INVENTORY-TYPE
               WHEN  TYPE-A-ITEM
                     PERFORM ITEM-A-REPORT
               WHEN  TYPE-B-ITEM
                     PERFORM ITEM-B-REPORT
               WHEN  TYPE-C-ITEM
                     IF VALUATION > SMALL-VALUE-TYPE-C
                        THEN
                            PERFORM SMALL-C-REPORT
                        ELSE
                            PERFORM LARGE-C-REPORT
                     END-IF
               WHEN  FASHION-ITEM
                     PERFORM FASHION-ITEM-REPORT
               WHEN  OTHER
                     PERFORM ITEM-ERROR-REPORT
            END-EVALUATE
    END-READ
 END-PERFORM.
```

This code is not without its problems. For one, most programmers would create a separate paragraph for the code listed in the EVALUATE. If we named that paragraph VALIDATE-MASTER, the code would read PERFORM VALIDATE-MASTER.

Scope Control

The earliest versions of COBOL employed two main methods of scope control:

- Punctuation in the form of the period (.)
- The beginning of a new procedure

Thus, in our COBOL-74 example of a nested IF, we saw the scope span all subsequent IF statements until the final period. In our COBOL-85 example, we nested statements using the END-IF scope terminator.

COBOL-85 introduces about 19 new scope terminators. These control the scope of iteration, selection, and sequential statements. For example, in an ADD statement we have

```
PERFORM WITH TEST BEFORE
  VARYING DEDUCTION-INDEX FROM 1 BY 1
  UNTIL DEDUCTION-INDEX > 20
        ADD  DEDUCTION (DEDUCTION-INDEX)
             TO  TOTAL-DEDUCTIONS
                 ROUNDED
             ON  SIZE ERROR
                 SET DEDUCTION-INDEX TO 21
                 MOVE "X" TO TOTAL-DEDUCTIONS-ERROR-FLAG
                 PERFORM ERROR-JOURNAL-ROUTINE
             NOT ON SIZE ERROR
                 IF TOTAL-DEDUCTIONS
                   > MAXIMUM-ALLOWABLE-DEDUCTIONS
                     PERFORM EXCESS-DEDUCTIONS-ROUTINE
                 ELSE
                     PERFORM DEDUCTION-ALLOWANCE-ROUTINE
                 END-IF
        END-ADD
END-PERFORM
```

Or, in a MULTIPLY:

```
 MULTIPLY DISCOUNT-RATE BY PURCHASE
    GIVING DISCOUNT
         ON SIZE ERROR
            PERFORM DISCOUNT-SIZE-ERROR
         NOT ON SIZE ERROR
            ADD 1 TO DISCOUNT-OK
 END-MULTIPLY
```

Once again, it would be unusual to code an ADD statement as robust as the one shown above. Under COBOL-85, not only is such code allowed, but it is readable. This example also illustrates a practical use of the in-line PERFORM ... END-PERFORM construct: table handling.

In COBOL-85, scope terminators are defined for verbs that have a built-in exception-handling facility such as the READ, SEARCH, or arithmetic verbs. In addition, you have probably noticed that some of the exception conditions now have negation available.

Subprograms

We now come to a subject that is difficult for many experienced COBOL programmers to grasp immediately, a concept that is foreign to the COBOL way. COBOL-85 now has both local and global variable capability. We can even explicitly define global variables with the word GLOBAL.

For those of you new to this computer language, COBOL has always had all data globally available in a program region known as the DATA DIVISION. Each and every procedure statement anywhere in the program could access any data item in that region; there was no way to hide it. Thus, you always knew where the data was—somewhere in the DATA DIVISION. However, global variables also resulted in the unfortunate lack of capability to create truly modular programs.

COBOL-85 has remedied this situation by implementing an expanded CALL verb. Although CALL has been available in COBOL-74 (and IBM's mainframe version of COBOL-74 was especially powerful), COBOL-85 allows us greater power in creating nested COBOL programs and accessing them via a CALL. Each CALLed

Continued on page 89

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Test Drive the COBOL Compiler Engineered for Companies

In the Fast Lane

Visual COBOL-85

GSA-certified, ANSI-85, native code COBOL compiler

MS-DOS · OS/2 · UNIX/386 · UNIX/68000 · CTOS/VM · Novell · IBM PC Network

---

Visual COBOL-85 is a high performance native code COBOL compiler that comes with its own integrated Screen Management System, which makes it easy to create a professional and dynamic user interface.

Visual COBOL-85 is a complete development environment that includes a multi-keyed ISAM facility, interactive debugger, internal and callable Sort options, extended CHAINing utilities, and much more.

Visual COBOL-85 provides interfaces to a variety of third party COBOL support tools, allowing you to expand your development environment as you expand your business.

Visual COBOL-85 requires no runtime fees on distributed COBOL applications, which can mean a savings of several thousand dollars.

Visual COBOL-85 applications are portable across a wide range of hardware platforms and operating environments.

Visual COBOL customers receive superb support from trained software engineers who understand your problems and are willing to help you solve them.

Visual COBOL-85 is backed by a company that has been providing COBOL solutions since 1968 and was the first to introduce a native code COBOL compiler for the IBM PC.

Visual COBOL-85 . . . find out about it. Contact MBP for more information and for details on a 30-day, no obligation "Test Drive" of the compiler.

For more information:

MBP Software and Systems USA
1131 Harbor Bay Parkway
Alameda, CA 94501

415-769-5333

Number 256 on the Reader Service Card

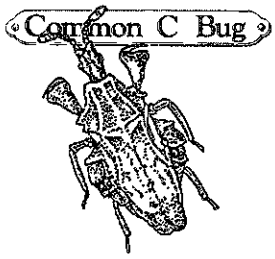
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

T-SHIRTS

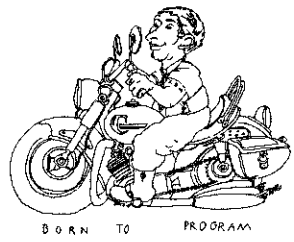
Multi-color imprints on the highest quality 50/50 cotton T-shirts.

$12.00 each — includes shipping & handling

SIZES S, M, L, XL



COMMON C BUG



BORN TO PROGRAM

PARALLEL PORTS

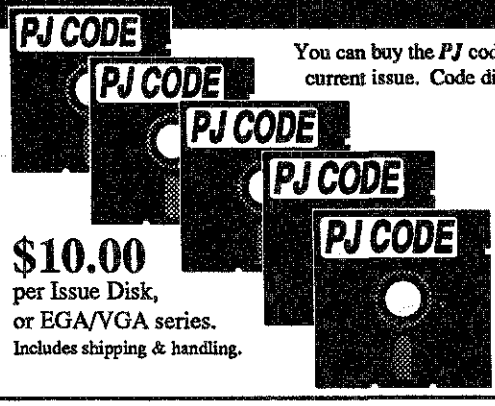
RS-232 HEAD

BUGS RULE 4 color imprint on a white shirt

All mail orders must be accompanied by check or Money Order — PREPAID ONLY.

CODE DISKS

Use the order form on the card opposite.
Order by phone with your VISA/MC.
CALL TOLL FREE
800/234-0386 ext 53



$10.00 per Issue Disk, or EGA/VGA series. Includes shipping & handling.

You can buy the *PJ* code listings from volume 5.5 to the current issue. Code disks for the Michael Abrash EGA/VGA series are also available.

Diskettes are available for the following issues:

5.5	6.4	EGA
5.6	6.5	VGA
6.1	6.6	
6.2	7.1	
6.3		

360K MS-DOS disks standard, other formats available

PJ ISSUE SPECIAL!

Back Issues ONLY $5.00 REG $7.00

BACK ISSUE SPECIAL

Back Issues Only $5.00!

Programmer's Journal celebrates 6 years service to the programming community.

Our birthday gift to you—any back issue for the low price of only $5.00 each.

Offer expires Feburary 28th, 1989

BACK ISSUES

$7.00 each PREPAID ONLY

A few of the Back Issues currently available

Vol. 4.6: DTACK GROUNDED; Unix Books, Part 1: How many amoebas at one o'clock; DOS functions 45H and 46H; DOS Terminate ans Stay Resident; Turbo Pascal inline code; Pascal—Assembler souce code for DOS function calls; Alignment problems in C and Managing memory allocation; Proposed ANSI C Language Standard, Part 4; The Canon A1 laser printer; Genifer dBASE applications generator reviewed; Advanced Trace86 debugger reviewed.

Vol. 5.2: DTACK GROUNDED; Starting your own software house; Pink noise; TSR graphics print screen utility; Coding guidelines for resident programs; Parallel processing with the EGA; C—portability in practice and using header files; Proposed ANSI C Language Standard, Part 6; Indexing in BASIC; A first look at Turbo C.

Vol. 5.3: Starting your own software house—publicity; Unix books, Part 2; Consulting—merger mania; EGA data control; Modula 2 for Pascal lovers; Optimizing Turbo Pascal; ANSI C language standard; Coping with ANSI C and forcing file updates; Formatting BASIC output with buffered PRINT USING.

Vol. 5.4: Starting your own software house—technical support; Unix books, Part 3; Consultant productivity and effectiveness; EGA set/reset capabilities; ANSI C language standard; Character tables and string functions in C; MWWIND—a menu with windows utility; Variable passing and hiding in QuickBASIC 2 subprograms.

Vol. 5.5: Intel's 80386 vs. OS/2; Metagraphics President Jack Davis interview; Consulting—making computer tools pay for themselves; ASM—the invisible CapsLock; TSRs using expanded memory with Turbo Pascal; Note on optimizing Turbo Pascal; Proposed ANSI C Language Standard; Printer output from Turbo C programs; Matchup of Turbo Basic 1.0 and QuickBASIC 3.0.

Vol. 5.6: Multics, OS/2, paging and segmentation; The Usenix conference and other news; How to get rich in consulting; Shell in Marshal Pascal; Turbo Pascal 4.0 preview; Handling fixed length charaacter arrays in C; The proposed ANSI C Language Standard.

Vol. 6.1: Consulting is an attitude; Write mode 3 of the VGA; The LIM 4.0 expanded memory specification; Environment variables in Turbo Pascal 4.0; A timer program in C; Porting mainframe software to the PC—FORTRAN compilers under DOS and the Defincon 32-bit coprocessing engines.

Vol. 6.2: Hal Hardenbergh on RISCs, CISCs, and Fabs; Pink noise; Yet another VGA write mode; Inside Turbo Pascal 4.0; The proposed ANSI C Language Standard; Converting a Turbo C program to a TSR; 80286 and 80386 programming; 80386 software development; The Norton Guide for BIOS/DOS Assembly.

Vol. 6.3: Hal Hardenbergh on memory trends; Pink noise; Getting your consulting tools in order; Gordon Letwin on OS/2 memory management; Virtualizing the 386; Pascal—A CASE of the jumps; Turbo Professional 4.0; BASIC—CALL, GOSUB, and multi-line functions

Vol. 6.4: Job-shopping; Windowing systems overview; Reading VGA memory; X Window—a view from DOS; A digital clock in C for MS Windows; Windowing in TopView, DESQview, & MS Windows; MS C 5.1 and the ANSI Standard; A mouse unit for Turbo Pascal 4.0.

Vol. 6.5: Hal Hardenbergh on computers in the fast lane; Consulting—no-win jobs and too much work; VGA text paging; Using dynamic arrays and records in Turbo Pascal 4.0; An OS/2 Presentation Manager application template; Watcom C 6.0 and the ANSI Standard; Object oriented OS/2 in C++; Ada and software engineering; REXX on the PC.

Vol. 6.6: PC DOS 4.0; OS/2 vs. Unix; VGA screen saving and color selection; MASM quirks; Addressing extra memory for DOS; DOS device drivers debugging and information utility; Proposed ANSI C Language Standard; Ray Duncan on Forth; Ada data types and exceptions; PC floating point hardware and software.

10% discount on any order of 10 or more T-Shirts, Disks or Back Issues.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

program may contain its own local data names and procedures. The general form is

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MAIN-PROGRAM.
          :
          :
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WS-PARAMETERS.
   05 MAIN-PARM1          PIC X(04).
   05 MAIN-PARM2          PIC X(12).
PROCEDURE DIVISION.
          :
          :
     CALL SUB-PROGRAM-1
          USING MAIN-PARM-1,
                MAIN-PARM-2
          :
          :
IDENTIFICATION DIVISION.
PROGRAM-ID. SUB-PROGRAM-1 IS INITIAL PROGRAM.
          :
          :
DATA DIVISION.
LINKAGE SECTION.
01 LINK-PARAMETERS.
   05 LINK-PARM1          PIC X(04).
   05 LINK-PARM2          PIC X(12).
PROCEDURE DIVISION
          USING BY CONTENT LINK-PARM1,
                           LINK-PARM2
          :
          :
EXIT-PARAGRAPH.
        EXIT PROGRAM.
```

In most implementations, the LINKAGE SECTION does not occupy storage space of its own. When SUB-PROGRAM-1 is CALLed by MAIN-PROGRAM in the above example, pointers are set up from the LINKAGE section back to the storage addresses of the corresponding parameters in the calling program. Although this is not quite as good as having separate storage areas for the named variables, it is better than what we have had in earlier COBOL.

The phrase "BY CONTENT" means that the parameters may not be modified in the storage area of the CALLing by the CALLed program. The default is that such values may be changed.

As in Pascal or FORTRAN, parameters are passed positionally; that is, the first parameter in the CALLing list must correspond in form and type to the first parameter in the USING list of the CALLed subprogram. When the subprogram CALL feature is combined with the COPY verb, designers have more capability for team programming on large projects. Each programmer can be working on a module independently of other members of the team. Even better, algorithms implemented for one project can easily be incorporated into new projects. For example, a compound-interest routine can be used over and over in many programs with no regard to the variable names. The programmer simply copies the routine into his or her program and CALLs it as a subprogram with appropriate parameters without being concerned with the underlying algorithmic implementation.

Some of you have read my article "Ada—a Software Engineering Tool" in an earlier article in *Programmer's Journal* (Volume 6.5, September/October 1988). COBOL subprograms implement some of the principles discussed in that article: information hiding, modularity, and localization. The hoped-for outcome is greater reliability, readability, modifiability, and understandability.

Optional Filler

One of my early COBOL acquaintances said that over 80 percent of the world is FILLER—clearly a pre-ecological observation. In COBOL-85, however, we can avoid the word FILLER when we choose to do so, without harmful effects. Therefore, the following:

```
01 MY-RECORD-LAYOUT.
   05 DATA-FIELD-1        PIC X(07).
   05                     PIC X(02).
   05 DATA-FIELD-2        PIC X(21).
   05                     PIC X(02).
   05 DATA-FIELD-3        PIC X(48).
```

is equivalent to

```
01 MY-RECORD-LAYOUT.
   05 DATA-FIELD-1        PIC X(07).
   05 FILLER              PIC X(02).
   05 DATA-FIELD-2        PIC X(21).
   05 FILLER              PIC X(02).
   05 DATA-FIELD-3        PIC X(48).
```

String Manipulation

One of the really frustrating elements of traditional COBOL has been its cumbersome string-handling capabilities. This function has been improved in COBOL-85 with features such as reference modification. Suppose we have string data defined as follows:

```
05 DATA-1    PIC X(12)   VALUE "ABCDEFGHIJKL".
05 DATA-2    PIC X(02)   VALUE SPACES.
05 DATA-3    PIC X(14)   VALUE "MNOPQRSTUVWXYZ"
05 DATA-4    PIC X(26)   VALUE SPACES.
```

We can do the following:

```
MOVE DATA-1 (7:2) TO DATA-2
```

with the result "GH" in DATA-2.

When this feature is used in combination with the STRING and UNSTRING commands already available in COBOL-74, we are able to do some powerful manipulation of string data. For example:

```
STRING DATA-1 DATA-3 INTO DATA-4
```

results in

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

in DATA-4.

New Level-88 Capabilities

The Level-88 condition names have been enhanced to allow "Boolean" operators TRUE or FALSE. Using an END-OF-FILE flag condition, we could have:

```
01 PROGRAM-CONDITIONS.
   05 END-OF-FILE-FLAG     PIC X VALUE "N".
      88 END-OF-FILE  VALUE "Y" FALSE "N".
```

In the PROCEDURE DIVISION, we would have:

```
OPEN INPUT ACCOUNT-MASTER-FILE
SET END-OF-FILE TO FALSE
PERFORM UNTIL END-OF-FILE
    READ ACCOUNT-MASTER-FILE
         AT END
              SET END-OF-FILE TO TRUE
    END-READ
END-PERFORM
```

Notice that the condition is altered with a SET command rather than with a MOVE.

The availability of a Boolean operator makes it possible to create COBOL decision structures that operate with greater clarity than before. In addition, selection statements, one of the elementary structures described by Bohm and Jacopini, are implemented more elegantly using Boolean operations.

Implementations

When COBOL was introduced in 1959, it was intended to be "open ended and capable of change and amendment." COBOL-85 is only the third revision to be approved by the American National Standards Institute (ANSI). Previous approved versions were COBOL-68 and COBOL-74. Many programming organizations still have not made the transition from COBOL-68 to COBOL-74. Some software developers provide source code with their products and keep the COBOL code compatible with all versions, including '68 and '74. These developers must decide what to do about COBOL-85. One

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

premier software vendor in Sunnyvale, California, uses the same COBOL for all computer platforms—mainframe, mini, micro. Consequently, the company must employ the most generic and primitive features of the language and cannot take advantage of COBOL-85's new features. Conversely, an in-house developer, such as Wells Fargo Bank, is able to plan projects with all the new COBOL-85 features because it is not sharing code with other organizations.

In the COBOL world, many managers learned their programming on the job. As these executives moved up the management ladder, their attention became focused more on business decisions and less on technical ideas. In addition, many managers have well-justified paranoia about introducing new software methods. (Many COBOL shops still do not believe in "structured programming.")

COBOL-85 is just beginning to find its way into production projects. IBM, which introduced a variant of COBOL-85 (named COBOL II) in 1984, has recently released its version of COBOL-85 for its mainframes. Many smaller vendors, such as Hewlett-Packard, Digital Equipment Corp., and PRIME, have had a version of COBOL-85 available for several years. Microsoft's earlier introduction of COBOL was a real disappointment (I wrote a small hospital project using it). To its credit, Microsoft has upgraded its COBOL commitment and is now offering a good version of COBOL-85. Also in the microcomputer world, REALIA has a good COBOL-85 compiler. I just finished teaching a COBOL class using Tandem Computers, and I'm really impressed with Tandem's implementation of COBOL-85. Also, HP's COBOL-85 under MPE-XL is nicely done.

OMNIVIEW

ROCK SOLID MULTITASKING!

Packed with the power you need, OMNIVIEW takes you beyond the limits of DOS.

- Time-slices multiple jobs
- Takes full advantage of EMS 4.0
- Optional support for '386 Systems
- Manages megabytes of programs
- Controls resident utilities
- Lets you cut and paste between programs

OMNIVIEW unleashes your PC & converts it into an easy-to-use, multitasking workstation. Whether you have an 80386 or 8088 based computer, OMNIVIEW will squeeze out every ounce of power.

Solid & bulletproof, it's 47K assembly language engine gives you all the power without added overhead or complexity. Just ask the reviewers.*

"Sunny Hill provides true multitasking for most PC applications. It performs remarkably well, causing almost no noticeable reduction in the performance of each program . . . The user manual is well organzied and easy to read . . . It is an excellent product."

PC Tech Journal Sept '87

"It does an amazing job of handling not only multitasking, but in many cases, concurrent processing as well. I was amazed at it's success in handling the "impossible" combinations I threw at it."

"It is one of the best designed & implemented programs I've ever worked with. It is also one of the most bulletproof."

"What sets it apart from the pack is the fact that it runs in a network environment."

Data Based Advisor Aug '87

OMNIVIEW requires an IBM PC, XT, AT or Jr compatible, and PC or MS DOS 2.x to 3.x. To get your copy, call toll free:

(800) 367-0651

or send $89.95 + 3.00 S&H ($8.00 Intl.) to:

Sunny Hill Software

PO Box 55278, Seattle, WA 98155-5278

For more information call (206) 367-0650 *Omniview formerly Taskview

DoubleDOS trademark Softlogic Solutions, Concurrent PC-DOS reg. trademark Digital Research Corp., MS Windows trademark Microsoft Corp., DESQview trademark Quarterdeck Office Systems, Inc. TopView trademark IBM Corp.

Number 85 on the Reader Service Card

The Future

As a programming language, COBOL has been criticized and praised; still, its future is bright. COBOL's new features make it useable for production programming under contemporary standards for software development. A huge literature exists in this language that a large number of programmers already know. One important characteristic of COBOL has always been its ease of learning and use. COBOL-85 makes it even easier. Moreover, college students in computer science programs are learning the kind of programming theory that is the very foundation of COBOL-85.

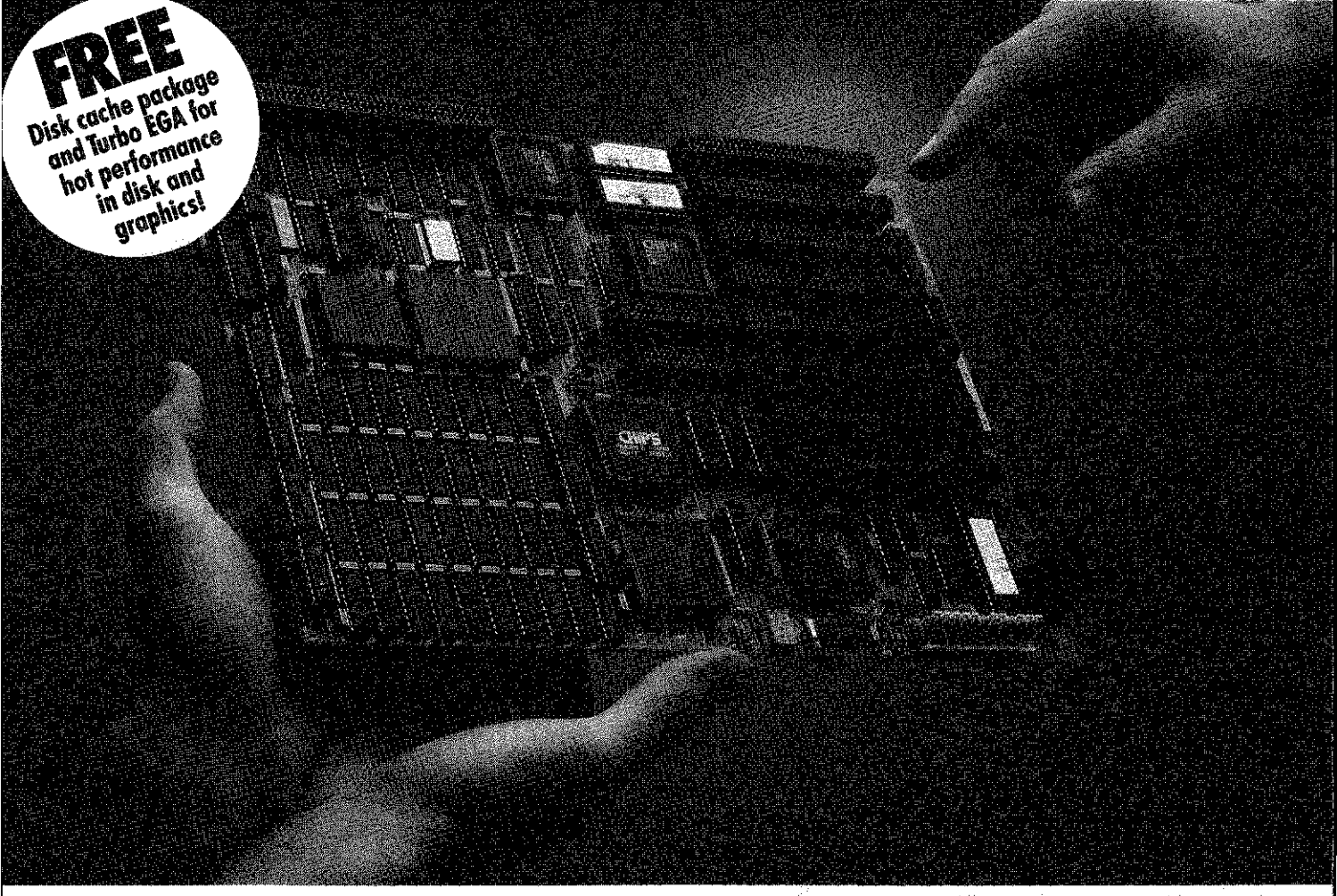
The strict program organization of COBOL makes it appropriate for use in environments that require adherence to well-defined standards. In addition, it can be used effectively by people with very little training or knowledge of computer science.

Manufacturers of computer equipment have made a commitment to on-going support of COBOL. Even with the advent of new methods of database management, COBOL has been able to keep up. In the IBM mainframe world, SQL and CICS are coded directly into the COBOL source code with statements such as EXEC SQL or EXEC CICS, where a COBOL preprocessor expands the embedded source code into statements useable by the COBOL compiler. IBM has gone even further with COBOL II (COBOL 85) and allowed its compiler actually to recognize the existence of CICS directly.

The language is not yet as good as it could be. My wish list includes a PERFORM statement that allows parameter passing, comments on the same line with executable code, additional exception-handling features, standard methods for on-line transaction processing, user-defined verbs, and greater subprogram independence. Still, the current definition for COBOL-85 brings the language a long way toward creating better software systems. Not only do we have better tools, but we can build on knowledge gained from the successful COBOL systems of the past. ♦

Richard D. Riehle is an instructor in Ada at Merritt College in Oakland, and Foothill College in Los Altos, California. He is also an independent computer consultant. Mr. Riehle's training organization ICE (for Intensive Computer Education) conducts seminars on a variety of computer subjects. Mr. Riehle has over twenty years in software development, including design, programming, and project management. He is currently working on a book, From COBOL to Ada, *a tutorial for experienced business programming professionals. Readers may contact Mr. Riehle at 1090-105 Tanland Drive, Palo Alto, CA 94303, (415) 858-1267.*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



CONVERT YOUR PC, XT OR AT INTO A HIGHER FORM OF LIFE!

386 MOTHERBOARDS FOR 386 SPEED.

Don't let your PC give up the ghost — Hauppauge has arrived with a new spark of life: the 386 MotherBoard™ Far more advanced than an accelerator card, our line of MotherBoards grace your PC, PC/XT, PC/AT or compatible with speeds equal to the IBM PS2 Model 80. And faster. Because we've built in 1 Megabyte of high speed RAM and a 387 math coprocessor socket for speeds that will make your PC glad to be alive.

UNIX/386 Compatible. To ensure a long, fruitful life, our 386 MotherBoards are compatible with the PC/AT (BIOS and I/O) — so you can run the new generation of DOS, OS/2. Our Boards also run Windows/386, UNIX V, PC-MOS/386, and XENIX/386. For *more* power, you'll find 16-bit expansion slots that accommodate the latest I/O expansion cards. Plus, we can add up to 15 Megabytes of extended memory. No 386 accelerator card can go over 2 Megabytes or give you so much versatility. **Only our 386 MotherBoards give your PC s an unlimited future!**

The Critics Applaud! *PC Magazine* awarded our Board "The Editor's Choice" for 386 Replacement Boards. *PC World* called it "the Upgrade Product of the Year."

Technical Features ▪ 16 or 20 MHz 80386 ▪ 1 Megabyte of 100 nsec 4-way interleaved RAM ▪ PC/AT compatible I/O and BIOS for support of OS/2 ▪ Six 8-bit expansion slots ▪ Two 16-bit expansion slots; four on 386 MotherBoard for AT ▪ One 32-bit expansion slot for up to 15 Megabytes of high speed memory ▪ Battery-powered clock calendar ▪ 2.9 MIPS at 16 MHz, 3.9 MIPS at 20 MHz.

386 MotherBoard for the PC or PC/XT	$1695
386 MotherBoard for the PC/AT	$1795
386 MotherBoard/20 MHz for PC/AT	$2195
32-bit RAM Board (2 Mbyte installed; up to 10 Mbytes)	$1195
16 MHz 80387 math coprocessor	$695
16-bit combination hard disk/floppy disk controller	$245

Prices subject to change.

For more information on our easy-to-install MotherBoard, call **1 (800) 443-6284**. In New York: **(516) 434-1600**. All our products are available at your local computer dealer!

Hauppauge Computer Works, Inc.
175 Commerce Drive,
Hauppauge, New York 11788.

Hauppauge!

Trademarks: IBM PC, XT, AT, PS2 and OS/2: IBM.
Intel 386: Intel Corp.; Windows/386: Microsoft Corp.

Number 122 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Programming Utilities

OPUS Make 5.1
OPUS Software
1032 Irving St., Suite 439
San Francisco, CA 94122
(415) 664-7901

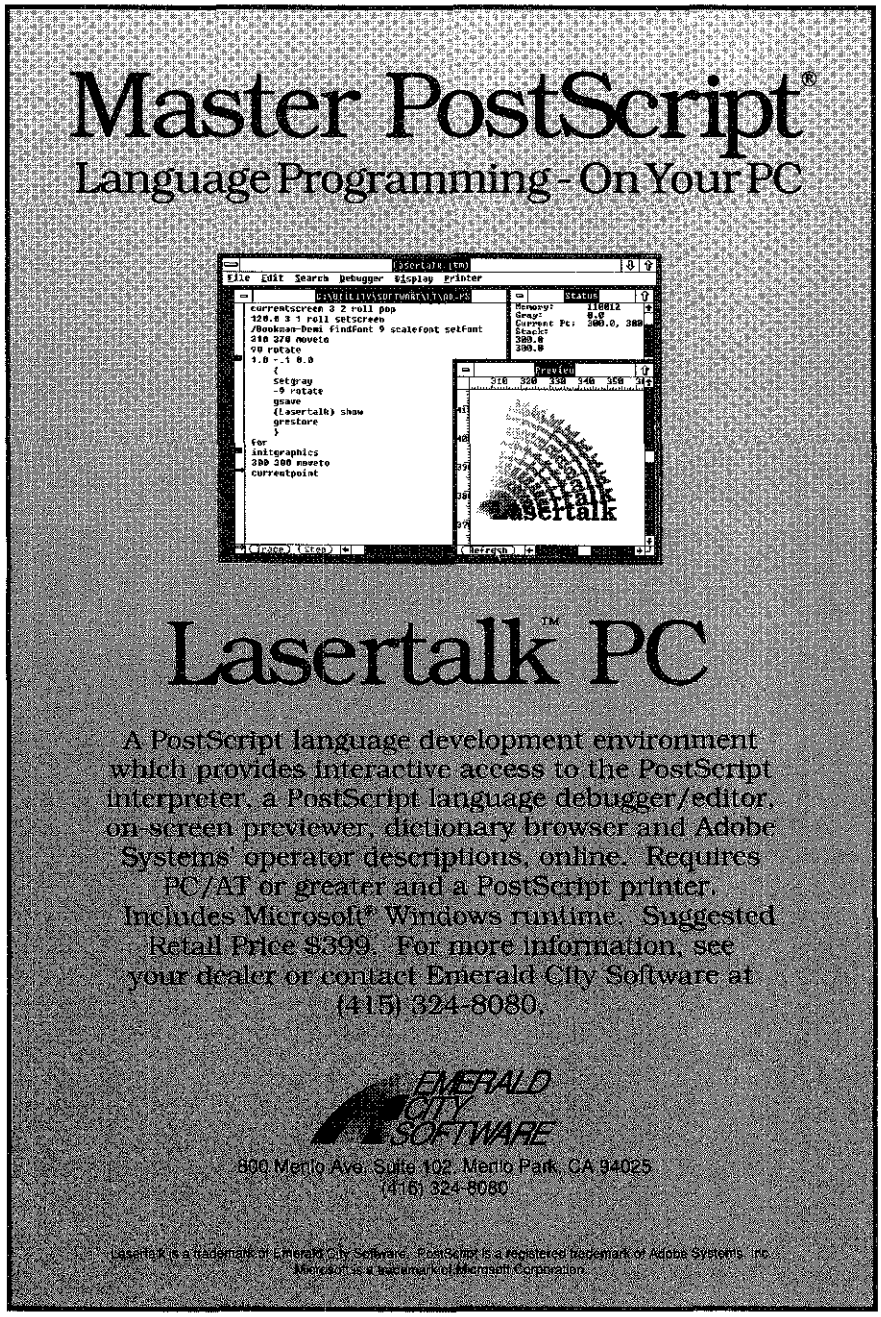
Continuing our look at make utilities, I've received a fine new product, OPUS Make Version 5.1. The OPUS Make package ($79) includes two versions of make for both DOS and OS/2 (small and large models), a program to create make dependency files, and several useful utilities. OPUS Make supports the Unix make dependency language with enhancements and extensions. OPUS Make executes quickly and under DOS supports an attractive memory-miser option. Rather than creating batch files, OPUS Make reduces the amount of memory it uses to a minimum by paging itself out. With this option, OPUS Make will extract less than 3K of overhead when executing your favorite compiler or linker yet still maintain full control. Another memory-reducing option is a one-command mode, which executes only one shell line from a makefile. This option is useful for doing compiles from inside editors, since it spares the overhead of paging, but it still reduces memory usage to about 1K. These features are unnecessary under OS/2.

OPUS Make supports 17 command-line options to control operation. Options may also be set via a predefined macro MFLAGS in a makefile or via an environment variable. These options include a directory snapshot feature to speed make execution; the DOS memory-miser features; a Microsoft compatibility mode; the ability to disable default inference rules, give environment macros higher precedence, and print information on macros, dependencies, and inference rules; and a debug/trace mode.

The Unix dependency file syntax has been extended with powerful loop and conditional directives similar to the C language. These directives begin with the ! character, much like extensions added by Borland to their make program. These directives support if, else, elif, endif, while, foreach, break, continue, end, include and error constructs. A rich set of predefined macros is provided. A VPATH macro specifies additional directory names to find files that don't have a path specification. Two special targets are supported: .BEFORE and .AFTER, which would be executed on initialization and after the Make process has been completed. This feature can be especially helpful in setting up an environment for different build processes and restoring the state at the process end. Other pseudotargets support the Polytron Version Control System logfiles; the creation of response files for Microsoft LINK, LIB, and other programs; control-C handling, and the issuance of warnings when using undefined macros.

Default inference rules are read at start-up from a MAKE.INI, if one exists. These default rules may be readily changed to support your current compiler/linker and build process. Under DOS, multiple commands in a single shell line are supported, separated by semicolons. File I/O redirection is supported for both stdout and stderr.

The package includes MKMF, a tool for generating and managing makefiles from user-defined templates. Dependency information is computed by scanning source files for included files. MKMF recognizes standard filename suffixes for C, C++, Pascal, FORTRAN, and assembler and can be extended. This utility works best for C and C++ files by understanding C preprocessor directives. Support for other languages is much weaker because of the limited "include" syntax supported. A useful TOUCH utility is also supplied to update files' date/time stamps or change them to some arbitrary time. The manual is on the terse side (47 pages—with no index—plus a 4-page addendum) and could clearly benefit from more examples. A


Master PostScript®
Language Programming - On Your PC
Lasertalk™ PC
A PostScript language development environment which provides interactive access to the PostScript interpreter, a PostScript language debugger/editor, on-screen previewer, dictionary browser and Adobe Systems' operator descriptions, online. Requires PC/AT or greater and a PostScript printer. Includes Microsoft® Windows runtime. Suggested Retail Price $399. For more information, see your dealer or contact Emerald City Software at (415) 324-8080.
EMERALD CITY SOFTWARE
800 Menlo Ave, Suite 102, Menlo Park, CA 94025
(415) 324-8080
Lasertalk is a trademark of Emerald City Software. PostScript is a registered trademark of Adobe Systems, Inc. Microsoft is a registered trademark of Microsoft Corporation.
Number 241 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

limited set of sample makefiles is provided, which could be enhanced with examples for compilers other than Microsoft and different languages. But if you are used to the Unix make syntax, OPUS Make appears to be a solid product.

BASIC Utilities

QuickPak Professional
Crescent Software
11 Grandview Avenue
Stamford, CT 06905
(203) 846-2500

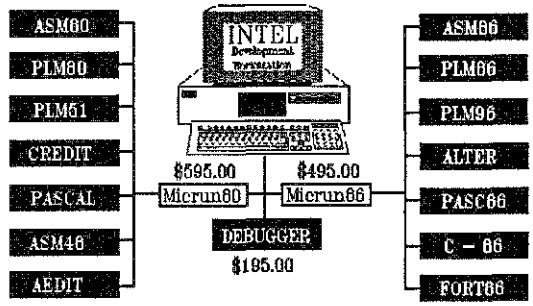
Over the last two years, I've witnessed a strong resurgence in popularity of BASIC. Clearly, the original PC language is learning some new tricks with fast new compilers from Microsoft and Borland. Crescent Software has released QuickPak Professional ($149), a comprehensive set of tools and utilities for BASIC programmers. The package consists of over 300 routines callable from BASIC, along with several dozen demonstration programs. Most of the routines are written in assembly language for speed and reduced size. The current release is available for use with Microsoft QuickBASIC 2, 3 and 4, BASCOM 6, and Borland's Turbo Basic. Complete BASIC and assembler source code is also provided.

All of the QuickBASIC 4 (QB4) data types are supported, including fixed-length and TYPE variables. The package includes extensive routines for DOS services (58), array functions (25), string manipulation (12), parsing and type conversion (17), scientific and financial functions (32), keyboard and mouse support (26), pull-down menu and input support (16), window management and video services (41), and numerous miscellaneous routines (49). A full set of file and printer routines is provided, eliminating the need to use BASIC's ON ERROR statements, with savings in executable file size and increased runtime speed. File and string encryption are supported, as are file and record locking. Screen dumps are supported in all graphics video modes to various printers. The video routines support multiple screen pages and EGA/VGA 43- and 50-line modes automatically.

The package also features a series of applications that may be added to your BASIC programs. These include a complete spreadsheet program; a full-screen editor with word wrap, block operations, and mouse support; a pop-up calculator; a calendar; an ASCII chart; and a file browser. The package comes on four disks, one with assembly language source. I found the BASIC and assembler source code files well documented, with extensive comments for use as a learning tool.

The documentation included is very extensive. The main QuickPak manual has more than 400 pages and contains a general discussion of the package, a separate page or two describing each function in detail, and a tutorial. Each routine is documented with a

RUN INTEL ON A PC



DEVELOP and DEBUG

8 and 16 Bit INTEL CODE ON YOUR PC...WITH

MICRUN

MICRUN 80....8 Bit Development Tool (6 or 10 MHZ)
MICRUN 86.....16 Bit Development Tool
8080 Debugger.....Intel Type Software Debugger

• Micrun 80/86 Package $ 895
(includes Micrun 80,86 & 8080 Debugger)

****

INTRODUCING
THE PC/INTEL DEVELOPMENT STATION

INCLUDES:
- Floppy Drive w/Controller
- MICRUN 80/86 PACKAGE
- MS-DOS & GW BASIC
- QUIKSAVE-Backup Utility Program

XT 10 MHZ V-20 $ 2,195

AT 12 MHZ 80286 $ 2,995

All systems ready to go. Pretested (48 hr burn-in). Optional equipment availble.

Call Today 1-800-888-8086

Micro Interfaces Corp. 16359 NW 57th Ave., Miami, FL. 33014

*Intel, Everex & Seagate are registered trademarks of their respective companies.

Number 214 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

brief description (purpose), variable description and syntax for both QuickBASIC 4/Bascom 6 and QB2/3/Turbo BASIC, an example of its use, and comments. The general sections describe software installation and use and provide a tutorial on passing variables to BASIC functions, linking with the supplied libraries, and building the libraries from the supplied assembler and BASIC source files. The appendix also functions as a tutorial, comparing various BASIC calling mechanisms (CALL, GOSUB, and multiline functions), random files, dynamic versus static arrays, long integers, and problems and pitfalls. In addition, a separate 60-plus-page tutorial is supplied for working with assembly language and BASIC. This tutorial is written in a light, humorous, and easily readable style. Unfortunately, neither of these manuals has an index.

I can heartily recommend QuickPak Professional to both the beginner (for its extensive tutorials) and the professional (for its breadth and depth of utilities). Upgrades from earlier QuickPak releases are available at nominal cost. Crescent Software also sells GraphPak Professional ($149)—a BASIC library of charting and graphics subroutines, LaserPak Professional ($149)—a set of BASIC subroutines for controlling HP Laserjet and various dot matrix printers, and other BASIC support products. Their QuickTALK catalog is published as an attractive typeset magazine, with articles and BASIC programming tips along with information on their products.

Standards

GOSIP
National Bureau of Standards (NBS)
B154 Technology
Gaithersburg, MD 20899
(301) 975-3240

For years, the TCP/IP protocols developed and supported by the Department of Defense have been the de facto standard for internetworking computers in government, military, and many private organizations, particularly among Unix machines. After a long-term effort supporting the open systems interconnect (OSI) networking standards development, the US government is finally moving ahead to implement the concept. The vehicle for this action is the recently adopted Government Open Systems Interconnection Profile (GOSIP)—don't you just love those acronyms?

GOSIP defines a common set of data communication protocols to enable systems developed by different vendors to interoperate and enable users of different applications on these systems to exchange information. GOSIP is based on the OSI protocols developed by the International Organization for Standards (ISO) along with agreements reached by the NBS and various vendors over the last two years. OSI defines a seven-layer network (cake) protocol. The GOSIP standard was officially approved in August 1988 and becomes effective in February 1989.

This standard mandates that new computer network products and services acquired by federal agencies must support GOSIP (OSI functionality). Currently, GOSIP supports Message Handling Systems and File Transfer, Access, and Management applications (FTAM). GOSIP also supports interconnection of the following network technologies: X.25; Carrier Sense Multiple Access with Collision Detection (CSMA/CD)—Ethernet and IEEE 802.3; Token Bus—IEEE 802.4; and Token Ring—IEEE 802.5. Starting in February 1989, agencies will be "encouraged" to acquire GOSIP-conforming new products. After 18 months, this will become a mandatory requirement in solicitations and contracts. Agencies also will be permitted to buy network products in addition to those specified in GOSIP. Both TCP/IP and OSI will likely coexist for some time on government networks.

NBS is offering the free publication *GOSIP Version 1.0* (June 1988), describing the sources of the protocol specifications, conformance, interoperability, performance testing, brief protocol descriptions, network addressing requirements, and security options. The detailed protocol specifications are contained in *Stable Implementation Agreements for Open Systems Interconnection Protocols* (NBS Special Publication 500-150). This document can be purchased for $12 from the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402; phone (202) 783-3238. Specify stock number 003-02838-0. ♦

USENIX and C++

by Vaughn Vernon

Until recently, learning C++ from the available publications was like trying to learn a foreign language from a Berlitz pocket guide. All of the right stuff is there (in minimal form), but for some reason you just can't stand the embarrassment of opening it and listening to yourself say those funny little words! Dunke shhh, what?

Fortunately USENIX, a nonprofit interest group of Unix users, has taken an active roll in promoting the C++ programming language. USENIX holds annual C++ conferences (last year's was in Denver). They also sell C++ Work Shop/Conference Proceedings from previous sessions in 1987 and 1988.

The Proceedings are 300 to 500 pages of leading-edge material on C++. The papers in these Proceedings are as important—or even more so— than the AT&T release notes on C++. For instance, the 1987 proceedings contain details on multiple inheritance (MI) in C++. MI is not yet available as part of a real product, and it may not be for a few more months. This presents two unique advantages to developers: First, you learn how to use a new feature well before it is available; second, you can warn C++'s designer and implementer, Bjarne Stroustrup at AT&T, of potential dangers before the feature is forever engraved in stone!

In any event, the conference Proceedings are the most up-to-date definition of C++. The Proceedings contain numerous case studies of C++ users and implementers as well as new tools designed to enhance C++'s usefulness and programmer productivity. I recommend that anyone having even a slight interest in C++ order both the 1987 and 1988 Proceedings.

The C++ conference itself is an invaluable four days of inspiring, intuitive sessions presented by the true experts in the field. The first two days are paid tutorials for newcomers. The conference itself is held on the next two days. A fifth day is devoted to implementers (only) of C++ translators and compilers.

Bjarne Stroustrup explains planned new features of C++ at the conference. In 1988, new features included Parameterized Types and Type-safe Linkage. Bjarne openly asks for constructive criticism of his ideas, which proves that he is indeed concerned about not introducing a faulty feature into C++ and forever infecting its users.

If I sound excited, it's because I am. The problem for many of you is that you will have to wait for 1989's C++ Conference. However, until then I urge you to get the C++ Conference Proceedings. Contact:

USENIX Association
P.O. Box 2299
Berkeley, CA 94710 USA

Proceedings: $20 per copy, prepaid. Phone orders are not available. Conference: $150 for two days. USENIX is a not-for-profit association. ♦

Vaughn Vernon is President of Aspen Scientific, makers of CURSES.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

UNDERSTANDING C VIDEO TAPE

Get guaranteed results with this three volume video course! For expert or ovice—these tapes (total 6 hours) over ANSI C for today's professional rogrammer. Loaded with examples, hese tapes cover it all! Handy eference book included.

$59.95 postpaid. Add appropriate tax in TX. Send check or money order to:

AUTOMATION EDUCATION
2309 Royce Drive
Arlington, TX 76016-1232
817-496-8747 **RS# 207**

HOT NEW UTILITY!

EXPERT KEY is a RAM-resident keyboard enhancement utility that learns words and phrases and later recognizes them after a few keystrokes. Using AI and Expert Systems techniques, it finds and pops up a smart prediction; you either tap the EXPERT KEY to complete the word or keep typing. Learns words, phrases, lines & editing sequences from the keyboard, screen or your personal dictionary. Highly configurable. Runs with your editor and applications. For PC/XT/AT/386.

$65 (Reg. $95) + $3 S&H, 30 day guar.

Newell Technologies, Inc.
27758 Santa Margarita Pwky. #330D
Mission Viejo, CA 92691
714-583-7848 **RS# 203**

AT LAST! A C SOURCE CODE ANALYSIS TOOL WITH FULL PREPROCESSING CAPABILITY

C:LINES/C:TREE features

- ANSI draft compatible preprocessor
- Outlined listings showing flow of control within procedures
- Tree diagrams showing flow of control between procedures
- Source code reformatter

$79.95, shipping $3 domestic, $15 overseas. MD residents add 5% sales tax. Requires PC compatible, 256K.

SoftRex
4807 Bethesda Ave. #287
Bethesda, MD 20814
301-881-8274 **RS# 185**

Correction

In the MISCELLANEOUS section of the November/December 1988 *PJ* (Volume 6.6), the name of the Seidl Make utility and the name of its maker, Seidl Computer Engineering, were misspelled in the review. We apologize for the error.

Computer & Technical
*** BOOKS ***
INFOSOURCE
International
PROGRAMMER'S PC SOURCEBOOK $24.95
TURBO C DEVELOPER'S LIBRARY $24.95
Box 238, Dept. PJ 108
Barryville, NY 12719–0238
1–800–327–7970

Number **217** on the Reader Service Card

SAVE DEBUGGING TIME

WATCH-I/O is a pop-up utility that shows names and technical info. of all files in use.

WATCH-I/O also shows name of last file opened, current program name, and last file not found.

WATCH-I/O is now only $24.95+ $5.00 S&H. VISA/MC, CK, MO.

Quantum Publishing
23860 Miles Road
Cleveland, OH 44128
800-342-6237 **RS# 201**

BASIC to C

BAS C v3.2 accepts BASICA, Quick BASIC, CBASIC86, or most other BASICs by adding new statements, functions(Language Extender). Generates structured, indented, scoped MS/Turbo/Lattice/Aztec C. All Memory model, compile option supported. Runs on MS-DOS, XENIX, UNIX. Demo disk. From $199.

GOTOLESS CONVERSION
P.O. Box 835910
Richardson, TX 75083
214-404-1404 **RS# 127**

PL/M TO C TRANSLATOR

LISP-based translator for major assistance in translating from INTEL PL/M to C language. Human readability a high priority. Sample listings available. Runs under MS-DOS 2 and up. Ports possible for VAX with VMS or BSD 4.2.

$5000 site license includes 3 months of telephone support.

Frank Korzeniewski Consulting
1564-A Fitzgerald Drive #137
Pinole, CA 94564
415-799-1819 **RS# 209**

C Language Training Seminars

Learn C language fundamentals.
4 days of hands-on, interactive coding sessions and lectures.
Two textbooks and over 25 source code programs provided.
Focus on Microsoft C and Turbo C.

Classes held monthly at The C-Team.
Corporate training on location performed.
Tuition: $995.

Call for information.

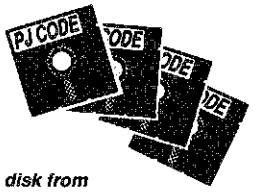
The C-Team, Inc.

Providing C expertise for all training and programming needs.

60 East 12th St.
New York, NY 10003
212/477-3990

Number **255** on the Reader Service Card

PJ Code Is Available!



On disk from

Programmer's Journal
P. O. Box 30160
Eugene, OR 97403
(800) 234-0386
(503) 747-0800 in OR

All code disks $10.00 each.
See ad on page 88.

PJ Code DiskScriptions are $39.95.
See ad on page 8.

On BBS from

Mnematics Videotex
P.O. Box 19
Sparkhill, NY 10976-0019
800-322-3633
914-365-0184 in NY

FIDO/OPUS Network
Blazer Net 152 Node 9
(503) 343-9724

Generation Five BBS
Voice 301-495-6690
Data 301-495-2932

Accessible via PC Pursuit, Washington, D.C. From Files submenu, select P for Personal Downloads.

PCS-49 on CompuServe
Section 9
Sponsored by:
MISOSYS, Inc.
P.O. Box 239
Sterling, VA 22170
Voice (800) MISOSYS

Turbo Source Search
P.O. Box 876
Scituate, MA 02066
Voice (617) 545-6677
Data (617) 545-9131

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ADVERTISER INDEX

Reader Service	Advertiser Name	Page Number
220	Aesoft Corporation	16
116	Aker	14
215	American Cosmotron, The	45
30	Austin Code Works	17
207	Automation Education	95
51	Az-Tech Software, Inc.	30
244	Block Island Technologies Inc.	77
3	Borland International	C4
255	C-Team, The	95
56	CompuView Products, Inc.	32
250	Courseware Applications, Inc.	26
229	Disk Software, Inc.	23
241	Emerald City Software	92
243	English Knowledge Systems	82
10	Essential Software	35
209	Frank Korzeniewski Consulting	95
*	Gimpel Software	79
127	Gotoless Conversion	95
248	Hammerly Computer Services, Inc.	66
122	Hauppauge Computer Works, Inc.	91
154	HavenTree Software LTD	46
217	Infosource International	95
226	Ithaca Street Software, Inc.	37
249	Jensen & Partners International	63
211	Kadak Products LTD.	57
63	Laboratory Microsystems, Inc.	57
44	Lahey Computer Systems, Inc.	53
246	Logical Alternatives, Inc.	29
212	Magna Carta Software	41
233	Mansfield Software Group	15
223	Marietta Systems	37
213	Max Software Consultants	51
256	MBP Cobol	87
230	McSoft, Inc.	58
247	Media Cybernetics	31
141	Merlin Publishing Group	40
143	Metagraphics Software Corporation	24
45	MetaWare, Inc.	1
214	Micro Interfaces Corporation	93
258	Microsoft	39
240	Milton Brown Software	57
*	PLEASE CONTACT THESE ADVERTISERS DIRECTLY	

Reader Service	Advertiser Name	Page Number
121	MISOSYS Inc.	51
238	MISOSYS Inc. Game Division	51
94	MMC AD Systems	78
109	NanoSoft	15
203	Newell Technologies, Inc.	95
160	Nu-mega Technologies	3
16	Opt-Tech Data Processing	59
254	Opus Software	41
195	Paradigm Systems	41
80	Paul Mace Software	26
208	PC Tech, Inc.	37
216	Peacock Systems	82
6	Periscope Company, The	70
26	Polytron Corporation	C2
*	*PJ Code DiskScription*	8
*	*Programmer's Journal* Mail Order Store	88
114	Programmer's Shop	9
228	Protech Marketing, Inc.	69
97	Qualstar Corporation	30
201	Quantum Publishing	95
252	Quantum Software	23
251	Radix Systems	26
222	Rainbow Technologies	23
20	Rational Systems, Inc.	81
198	Research Group, The	47
148	SemWare	34
165	SLR Systems	36
33	Soft Advances	22
185	SoftRex	95
98	Software Engineering Store, The	71
221	Software Security	5
257	Sophisticated Software	49
158	Summit Information Systems, Inc.	53
85	Sunny Hill Software	90
253	Sutrasoft	21
166	Tool Makers, The	53
25	TurboPower Software	44
125	V-Communications	11
100	Western Wares	77
237	XLN Software	6
182	Zortech, Inc.	C3
*	PLEASE CONTACT THESE ADVERTISERS DIRECTLY	

When contacting any of the above advertisers, please tell them you saw their ad in ***Programmer's Journal*** and that you appreciate their continuing support of this magazine.

Programmer's Journal

ADVERTISE IN

March/April 1989 Issue
Volume 7.2

Space Reservation Deadline:
January 6, 1989
Materials Due:
January 20, 1989

Call Programmer's Journal Ad Department to reserve your space.
(503) 747-0800

COMING UP—

March/April
Operating Systems

May/June
Database Coverage

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

A GP command sequence is like a program, but in a very simple language. You load a string of GP commands into the 82786 memory, terminate it with an NOP that has its ECL (End of Command Line) bit set, and then tell the GP to start executing from a specific point in memory. Since the chip does not have any conditional branch instructions, its programmability is severely limited. It does have CALL and RETURN commands, though; so you could set up a generic command list to, for example, draw a box with an *X* in it at the current *x,y* position and make it a subroutine that you could call in from your 82786 "program."

Table 1. Logical Writing/Pixel Processing Mode on the 34010 (The first 16 are also supported by the 82786.)

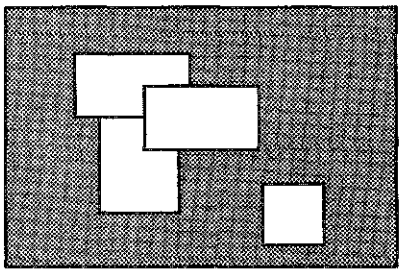
Definitions:

0s	All bits in the pixel are set to 0
1s	All bits in the pixel are set to 1
DST	The pixel at the destination
SRC	The source pixel
AND	Bitwise AND operation
OR	Bitwise OR operation
XOR	Bitwise XOR operation
NOT	Bitwise NOT (one's complement) operation
ADD	Unsigned Addition
SUB	Unsigned Subtraction
ADDS	ADD with Saturation (no wrap, max's out at max pixel value)
SUBS	SUB with Saturation (no wrap, stops at a pixel value of 0)
MAX	Larger of the two operands
MIN	Smaller of the two operands

The result of a pixel processing operation is placed in DST.

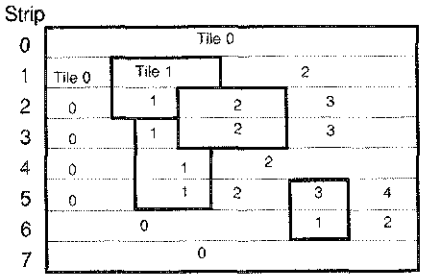
Processing Modes:

```
SRC
SRC AND DST
SRC AND (NOT DST)
0s
SRC OR (NOT DST)
(NOT SRC) XOR DST
NOT DST
(NOT SRC) OR (NOT DST)
SRC OR DST
DST
SRC XOR DST
(NOT SRC) AND DST
1s
(NOT SRC) OR DST
(NOT SRC) AND (NOT DST)
NOT SRC
SRC ADD DST
SRC ADDS DST
DST SUB SRC
DST SUBS SRC
MAX(SRC, DST)
MIN(SRC, DST)
```



Shaded area is field, i.e., not data.

What you see on screen, 4 hardware windows



Strip and tile breakdown of an 82786 screen

Hardware windows are similar in concept to the overlapping software windows utilized in such packages as Microsoft Windows. The big difference is that hardware windows do not require a redraw of screen information when the window is moved. This means that you can have virtually simultaneous drawing going on in a number of overlapping windows without worrying about clipping, because the hardware windows only display what's in memory; they do not alter memory. Hardware windows can also display *field*, a term used to indicate non–memory consuming display information ("dead space"). Because all normal limitations are removed, the long-time tie between memory and display is completely altered, requiring programmers to toss out everything they know about displaying PC-based graphics.

Of course, you can still use the one-to-one correlation between memory and display by creating a full-screen hardware window. Hardware windows are rectangular tiles that display data from memory at 1, 2, 4, or 8 bits per pixel. To top that off, a window can be flagged to indicate to the hardware that it should be zoomed. The zoom factor ranges from 1 to 64 in both the *x* and *y* directions. The screen position of a hardware window as well as the contents are determined by the DP display list. This display list consists of a series of strips, each of which contains a number of tiles. There is a limit of 15 tiles per strip, but no limit on the number of strips, providing each one is at least one scan line high.

Although the 82786 is a coprocessor, it does have three sets of registers that can be accessed by an external processor. One set belongs to the DP and is used for setting display frequencies, display list location, cursor pattern, and so forth. Another set belongs to the GP and contains information regarding the current drawing environment, such as *x,y* position, colors, and start of drawing area in memory. The last set of registers is the set containing the processor control. This set can be either memory

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Programmer's Journal
The Resource Journal for IBM Micro Programmers

P. O. Box 30160 • Eugene, OR 97403 • **(503) 747-0800**

January 5, 1989

Dear *PJ* Reader:

As you've probably noticed by now, page 38 of the January/February 1989 *Programmer's Journal* is missing a column of text. Mistakes like this are easily and often blamed on "printer error." Well, this time it *wasn't* our fault: Our printer really did goof, and they are working with us to make it up to you.

Although SLR Systems is no doubt thrilled to have their ad appear twice in one issue, we're sure you would rather have the material from Jake Richter's article. Therefore, we are having the printer send you a copy of the corrected page on the back of this letter. (If it's not there, they are *really* in trouble.)

We apologize for any inconvenience this blunder may have caused you. If we're all lucky, however, you may have a collector's item on your hands: The issue may contain the last mistake you ever find in *PJ*. (I know . . . and pigs may learn to fly in 1989, too.)

Sincerely yours,

Robert E. Wanetick

Robert E. Wanetick
Managing Editor

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LAST CHANCE AT

$99.95

EXPIRES FEBRUARY 1
Regular price $149.95.

WALTER BRIGHT – Programmer

"This is a revolutionary product which reflects the fact that Walter Bright is one of the world's best programmers."
Andy Redfern, Personal Computer World

"I find it amazing that Walter continues to find ways to improve his already excellent product."
Scott Robert Ladd, Micro Cornucopia

FROM ZORTECH INC.

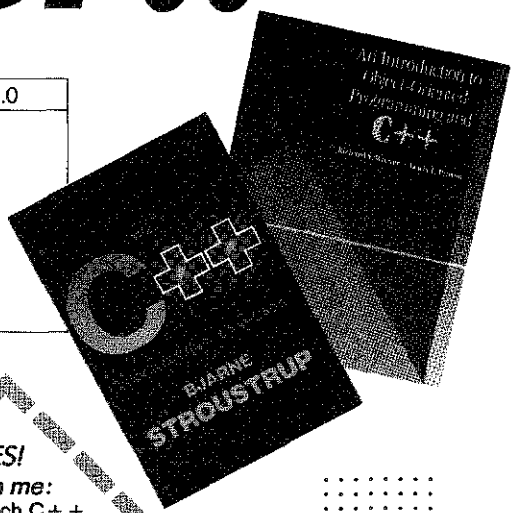
C is great!
C++ is better!!

Performance benchmarks

Benchmark	Zortech C	Zortech C++	Turbo C 1.5	Quick C 1.0
Sieve	20.49	20.54	23.62	22.72
Rsieve	20.49	20.54	23.62	22.03
Integer	1.32	1.38	6.31	6.49
Float	0.17	0.22	52.29	51.03
*Float	32.73	37.74	52.39	51.63
Pointer	17.91	17.96	17.13	16.87
Rpointer	17.79	17.91	17.14	16.64
Loop	3.90	3.90	3.90	3.90
Optimize	0.49	0.60	8.46	8.79

Benchmarks were run on an 80286 based IBM compatible at 6Mhz with no 8087.

The float benchmark was re-run without optimization.



Zortech C++ is the first native-code C++ compiler for MS-DOS compatible machines. This means you don't need to spend a lot of money on a C compiler. In fact, Zortech C++ comes with a C++ compiler, a C compiler, a linker, a librarian, a fully integrated editing environment, context sensitive help and the fastest graphics library you've ever seen!

With Zortech C++, you can mix and match code in C++, C and assembler and link them all together with a single command!

Zortech C++ comes with a complete C++ and C library, including compatibility with the ANSI C draft. Unique to Zortech C++ is support for mice, fast screen writes, re-entrant floating point and speaker support.

Now you can move your Microsoft and Turbo C code over to Zortech C++ – we've included compatible library functions! Also, we have made Zortech C++ compatible with Microsoft's Codeview debugger!

Enter the world of Object Oriented Programming (oops) with Zortech C++!

Sure, C is great – but C++ is better!

NOW AVAILABLE...
Std. Library Source Code
*Only $49.95 * (with C++ Purchase)*

YES!
Rush me:

- ☐ Zortech C++
- ☐ Library Source $49.95*
- ☐ Intro to OOP and C++ $27.95*
- ☐ The C++ Programming Language $29.95

VISA/MC/COD CHECK ACCEPTED

ZORTECH

Name.....

Address.....

..... *Phone*.....

VISA or MC#..... *Exp. date*.....

To: ZORTECH INC. 366 Massachusetts Ave., Arlington, MA 02174. Tel: 617-646-6703. Fax: 617-643-7969.

CALL THE ORDER HOTLINE 1-800-848-8408

Number 182 on the Reader Service Card. @ 1/2

* Please add $5.05 shipping to your order (UPS Ground)

REPRODUCED BY UMI.ORG
ELECTRONIC REPRODUCTION PROHIBITED

NOW SHIPPIN

Introducing Turbo Debugger, Turbo Assembler, Turbo Pascal 5, Turbo C 2: The Revolution Continues!

TURBO ASSEMBLER® AND TURBO DEBUGGER®

THE NEXT GENERATION

NEW

The new Turbo Assembler lets you write the tightest, fastest code. We used Turbo Assembler to write best sellers such as Quattro* and Turbo Pascal; here's why:

BGIDEMO BENCHMARK	TURBO ASSEMBLER	Microsoft* Assembler
Assembly time (seconds)	9.34	27.46
Link time (seconds)	4.15	10.51
FEATURE COMPARISON		
MASM compatible (4.0, 5.0, 5.1)	Yes	No
Thorough type checking	Yes	No
Nested structures and unions	Yes	No
Multimodule cross reference	Yes	No
Assemble multiple files	Yes	No

Run on IBM PS/2 model 60 using Turbo Assembler version 1.0, Turbo Linker version 2.0, Microsoft Macro Assembler version 5.10, Microsoft Overlay Linker version 3.64.

- Faster than other assemblers
- MASM compatible (4.0, 5.0, and 5.1)
- Significant new assembly language extensions
- Easy interfacing with high-level languages *including Turbo C and Turbo Pascal*
- Full 386 support

TURBO ASSEMBLER & TURBO DEBUGGER BOTH TOGETHER ONLY $149.95

The Turbo Debugger is truly revolutionary. It's the professional productivity booster that you've been waiting for. Check out the features:

FEATURE COMPARISON	TURBO DEBUGGER	CodeView*
Multiple overlapping views	Yes	No
386 virtual-86 mode debugging	Yes	No
Remote debugging	Yes	No
Data debugging	Yes	Partial
Generalized breakpoints	Yes	No
Session logging	Yes	No
Conventional memory used—80386	Zero K	230K
Conventional memory used—remote	15K	N/A

Turbo Debugger version 1.0, Microsoft CodeView version 2.2.

DATA DEBUGGER

- Follow pointers through linked lists
- Browse through arrays and data structures
- Change data values

DEBUG ANY PROGRAM

- Turbo Pascal, Turbo C, Turbo Assembler
- EMS support
- 386 virtual machine and remote machine debugging
- Supports CodeView* and .MAP-compatible programs

BREAKPOINTS

- Actions: stop, run code, log expression
- Break on condition, memory changed
- Software ICE capabilities
- 386 debug register support
- Support for hardware debuggers

TURBO PASCAL® 5.0

COMPILE, DEBUG AND RUN!

NEW

Turbo Pascal, the fastest programming language around and the worldwide favorite, is now better than ever with a sophisticated debugging environment right at source level, plus a separate command-line compiler.

FEATURE COMPARISON	TURBO PASCAL 5.0	Turbo Pascal 4.0
Integrated debugger	Yes	No
Overlays, including EMS support	Yes	No
8087 floating-point emulation	Yes	No
Turbo Debugger support	Yes	No
Procedural types, variables, parameters	Yes	No
Smart linking of code and data	Yes	No
Constant expressions	Yes	No
EMS support for editor	Yes	No

NEW TURBO PASCAL 5.0

- Compile, edit and debug in the integrated Turbo Pascal environment
- Integrated source-level debugger lets you step code, watch variables, and set breakpoints
- Overlays, including EMS support
- 8087 floating-point emulation
- Support for Borland's Turbo Assembler and Turbo Debugger
- Procedural types, variables and parameters
- Smaller, tighter programs: Smart Linker strips unused code and data
- Constant expressions
- EMS support for editor
- Only $149.95

TURBO PASCAL PROFESSIONAL SERIES: TURBO PASCAL 5.0 PLUS BOTH TURBO ASSEMBLER AND TURBO DEBUGGER—ALL THREE FOR ONLY $250!

Turbo Pascal 5.0, *plus* the new Turbo Debugger, the new definition of the state of the art in source-level debugging. *And* the new Turbo Assembler, the fastest assembler ever, including special directives for easy interfacing to Turbo Pascal. Only $250.

TURBO C® 2.0

NEW

THE PROFESSIONAL'S CHOICE

Turbo C is the professional's no-compromise choice. We at Borland used Turbo C to write industry best-sellers such as Sprint* and Reflex.*

FEATURE COMPARISON	TURBO C 2.0	Microsoft* C 5.1
Integrated debugger	Yes	No*
Inline assembly	Yes	No
Auto dependency checking	Yes	No
EMS support for edit buffer	Yes	No
Device-independent graphics	Yes	No
Number of memory models	6	5
Price	$149.95	$450.00

*Integrated debugger included with Quick C.

NEW TURBO C 2.0

- Compile, edit and debug in the integrated Turbo C environment
- Supports 6 memory models
- Over 430 library functions, including a complete graphics library
- MAKE facility with automatic dependency checking
- Step code, watch variables, and set breakpoints in the integrated source-level debugger
- Only $149.95

NEW TURBO C PROFESSIONAL SERIES: TURBO C 2.0 PLUS BOTH TURBO ASSEMBLER AND TURBO DEBUGGER—ONLY $250!

Professional Turbo C gives you the tools you need to write and debug professional-quality programs in optimized C and Assembly language. You are provided with Turbo C's own integrated development environment, plus Turbo Assembler and the standalone Turbo Debugger. Only $250.

HOW TO UPGRADE NOW!

TURBO C AND TURBO PASCAL OWNERS UPGRADE NOW!

If you're a registered Turbo C or Turbo Pascal owner, you can upgrade and get the latest version of your favorite language. The upgrade manual plus disks for Turbo C 2.0 and Turbo Pascal 5.0 are yours for just $49.95 (plus tax if applicable and $5.00 for handling within the U.S.; $10.00 in Canada).

CALL NOW AND BE SURE TO HAVE YOUR TURBO C OR TURBO PASCAL SERIAL NUMBER HANDY.
(408) 438-5300

AVAILABLE AT YOUR LOCAL DEALER OR CALL (408) 438-5300 TO UPGRADE

BORLAND

Number 3 on theReader Service Card

BI 1306

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED